instances of the application execute on a plurality of servers, the method comprising:

receiving a client request for an application at a first server operating a first instance of the application;

executing a first status module on the first server, wherein said first status module is configured to determine a first status of the first instance;

executing a server monitor module on the first server, wherein said server monitor module is configured to receive a first status of a second instance of the application operating on a second server;

examining said first status of the first instance and said first status of the second instance to select a preferred server from among the plurality of servers; and

routing the client request to said preferred server.

L17: Entry 12 of 14 File: USPT Feb 23, 1993

DOCUMENT-IDENTIFIER: US 5189667 A
TITLE: Method and apparatus for controlling call processing based upon load conditions

Abstract Text (1):
In a call set-up control operation used for a data packet communication network, a call set-up controlling method comprises the steps of: preparing a plurality of call set-up algorithms different from each other; and, selecting one of the call set-up algorithms suitable for a call set-up demand, depending upon at least a load condition of the packet data communication network, whereby the call set-up operation is carried out in accordance with the selected call set-up algorithm.

Application Filing Date (1):
19910110

Brief Summary Text (12):
selecting one of the call set-up algorithms suitable for a call set-up demand, depending upon at least a load condition of the packet data communication network, whereby the call set-up operation is carried out in accordance with the selected call set-up algorithm.

CLAIMS:

1. A method for controlling call set-up operation in a packet data communication network comprising the steps of:

preparing a plurality of call set-up algorithms different from each other; and

selecting one call set-up algorithm of said call set-up algorithms suitable for a call set-up demand, depending upon at least a load condition of said packet data communication network, whereby said call set-up operation is carried out in accordance with said one call set-up algorithm selected.

13. A packet transfer control apparatus as claimed in claim 9, wherein said judging means judges said priority orders of said data packets based upon not only said load condition of said data packet communication network, but also upon a quantity of calls waiting to be processed.

First Hit Fwd Refs

Generate Collection

L9: Entry 17 of 24 File: USPT Oct 3, 2000

DOCUMENT-IDENTIFIER: US 6128279 A
TITLE: System for balancing loads among network servers

Application Filing Date (1):
19981001

Brief Summary Text (6):
While round-robin DNSs can coarsely distribute loads among several servers, they have several drawbacks. For example, not all requests for connection to a Web site are necessarily received by a round-robin DNS. Rather, many requests will have been previously "resolved" by a DNS local to the requester and remote from the Web site (i.e., a "a remote DNS") or by the requester (i.e., the computer that issued the request on the WWW). In these cases, resolution is based on an address which has been cached in the remote DNS or the requestor, rather than by sequential rotation provided by the Web site's round-robin DNS. Due to this caching, load balancing may not be achieved to a satisfactory degree.

Brief Summary Text (8):
As an alternative to the DNS-based load balancing techniques described above, some vendors have introduced dedicated load balancing hardware devices into their systems. One such system includes a device, called a proxy gateway, which receives all network requests and routes those requests to appropriate Web servers. In particular, the proxy gateway queries the servers to determine their respective loads and distributes network requests accordingly. Responses from the servers are routed back to the network through the proxy gateway. Unlike the DNS-based schemes, all requests resolve to the IP address of the proxy server, thereby avoiding the risk that remote DNS caching or failed servers will inadvertently thwart access to the site.

Brief Summary Text (13):
Dedicated load balancers, such as proxy gateways and IP redirectors, also have problems when it comes to electronic commerce transactions. In this regard, electronic commerce transactions are characterized by multiple sequential requests from a single client, where each subsequent request may need to refer to state information provided in an earlier request. Examples of this state information include passwords, credit card numbers, and purchase selections.

Detailed Description Text (8):
Internal network 12 includes mainframe 16 and back-end Web servers 27 and 29. Back-end Web servers 27 and 29 comprise file servers which store a database for Web site 1. Back-end Web servers 27 and 29 may be used to access data files on mainframe 16 (or other similar computer) in response to requests from server cluster 6. Once such data files have been accessed, mainframe 16 may then transmit these files back to server cluster 6. Alternatively, data on back-end Web servers 27 and 29 may be accessed directly from server cluster 6 without the aid of mainframe 16.

Detailed Description Text (10):
FIG. 2 illustrates process steps of the present invention for load balancing received network requests. To begin, in step S201 a network request is received at a server, such as server 7 show in FIG. 3. This request may be resolved by a remote

DNS on the Internet based on a cached IP address (e.g., requests 1, 2, 3 and 4) or, alternatively, the request may be resolved by a local round-robin DNS 4 (e.g., request 5). Then, in step S202, server 7 determines a load (e.g., the number and/or complexity of network requests) that it is currently processing, and the capacity remaining therein.

Detailed Description Text (12):
If step S203 decides that server 7 is not processing a load that exceeds the first predetermined level, flow proceeds to step S204. In step S204, the network request is processed in server 7, and a response thereto is output via the appropriate channels. On the other hand, in a case that step S203 determines that server 7 is processing a load that exceeds the first predetermined level, flow proceeds to step S205.

Detailed Description Text (13):
Step S205 determines loads currently being processed by server 7's peers (e.g., servers 9 and 10 shown in FIG. 3). In more detail, in step S205, load balancing module 17 compares its current load information with the most recent load information provided by load balancing modules 19 and 20. These load balancing modules continuously exchange information regarding their respective loads, so that this information is instantly available for comparison. In the example shown in FIG. 3, load balancing module 19 provides information concerning the load currently being processed by server 9, and load balancing module 20 provides information concerning the load currently being processed by server 10.

Detailed Description Text (14):
In step S206, load balancing module 17 determines whether the loads currently being processed by server 7's peers are less than the load on server 7 by a differential exceeding a second predetermined level. In preferred embodiments of the invention, this second predetermined level is 20%, which provides a means of assessing whether servers 9 or 10 have at least 20% more of their capacities available than server 7. Of course, the invention is not limited to using 20% as the second predetermined level. In this regard, as above, a value for the second predetermined level may be stored in a memory on server 7, and may be reprogrammed periodically.

Detailed Description Text (17):
Step S207 determines which, if any, of the servers at Web site 1 are off-line based, e.g., on the load information exchange (or lack thereof) in step S205. A server may be off-line for a number of reasons. For example, the server may be powered-down, malfunctioning, etc. In such cases, the servers' load balancing modules may be unable to respond to a request from load balancing module 17 or otherwise be unable to participate in an exchange of information, thereby indicating that those servers are off-line. In addition, in preferred embodiments of the invention, the load balancing modules are able to perform diagnostics on their respective servers. Such diagnostics test operation of the servers. In a case that a server is not operating properly, the server's load balancing module may provide an indication to load balancing module 17 that network requests should not be routed to that server.

Detailed Description Text (18):
Next, step S208 analyzes load information from on-line servers in order to determine which of the on-line servers is processing the smallest load. Step S208 does this by comparing the various loads being processed by other servers 9 and 10 (assuming that both are on-line). Step S209 then routes the network request to the server which is currently processing the smallest load. In the invention, routing is performed by sending a command from load balancing module 17 to a requestor instructing the requestor to send the request to a designated server. Thus, re-routing is processed automatically by the requestor software and is virtually invisible to the actual Internet user.

Detailed Description Text (20):
FIG. 3 illustrates load distribution according to the present invention. More specifically, as noted above, server 7 (more specifically, load balancing module 17) receives requests 1, 2, 3 and 4 resolved by network DNS 21 and request 5 via local DNS 4. Similarly, server 10 receives request 6 (i.e., a cached request) via local DNS 4. Any of these requests may be "boolanarked" requests, meaning that they are specifically addressed to one server. Once each load balancing module receives a request, it determines whether to process that request in its associated server or to route that request to another server. This is done in the manner shown in FIG. 2. By virtue of the processing shown in FIG. 2, load balancing modules 17, 19 and 20 distribute requests so that server 7 processes requests 1 and 2, server 9 processes requests 3 and 5, and server 10 processes requests 4 and 6.

Detailed Description Text (30):
In this regard, except for proxy 26, the features show in FIG. 5 are identical in both structure and function to those shown in FIG. 1. With respect to proxy 26, proxy 26 is used to receive network requests and to route those requests to appropriate servers. A load balancing module on each server then determines whether the server can process requests routed by proxy 26 or whether such requests should be routed to one of its peers. The process for doing this is set forth in the first, second and third embodiments described above.

CLAIMS:

1. A method of distributing requests among a plurality of network servers, the method comprising the steps of:

receiving a request from a remote source at a first one of the network servers;

executing a determining step in the first server, the determining step for determining whether to process the request in the first network server;

processing the request in the first network server in a case that the determining step determines that the request should be processed in the first network server; and

routing the request to another network server in a case that the determining step determines that the request should not be processed in the first network server;

wherein the determining step comprises the steps of:

determining a load currently being processed by the first network server; and

receiving information in the first network server from each of the other network servers, the information from each of the other network servers comprising information concerning a load currently being processed in each network server;

wherein the determining step determines that the first network server should process the request in a case that (i) the load currently being processed in the first network server is below a first predetermined level, or (ii) the load currently being processed in the first network server is above the first predetermined level and is above loads currently being processed by either of the other network servers by less than a second predetermined level; and

wherein the determining step determines that the first network server should not process the request in a case that the load currently being processed in the first network server is above the first predetermined level and a load currently being processed in at least one of the other network servers is below the level of the first network server by at least the second predetermined level.

2. Computer-executable process steps stored on a computer-readable medium, the computer executable process steps comprising a server module which is installable in a plurality of network servers to distribute requests among the plurality of network servers, the computer-executable process steps comprising:

code to receive a request from a remote source at a first one of the network servers;

code, executable by the first server, to determine whether to process the request in that server;

code to process the request in the first network server in a case that the determining code determines that the request should be processed in the first network server; and

code to route the request to another network server in a case that the determining code determines that the request should not be processed in the first network server;

wherein the determining code comprises:

code to determine a load currently being processed by the first network server; and

code to receive information in the first network server from each of the other network servers, the information from each of the other network servers comprising information concerning a load currently being processed in each network server;

wherein the determining code determines that the first network server should process the request in a case that (i) the load currently being processed in the first network server is below a first predetermined level, or (ii) the load currently being processed in the first network server is above the first predetermined level and is above loads currently being processed by either of the other network servers by less than a second predetermined level; and

wherein the determining code determines that the first network server should not process the request in a case that the load currently being processed in the first network server is above the first predetermined level and a load currently being processed in at least one of the other network servers is below the level of the first network server by at least the second predetermined level.

3. A network server which is capable of processing requests and of distributing the requests among a plurality of other network servers, the network server comprising:

a memory which stores a module comprised of computer-executable process steps; and

a processor which executes the process steps stored in the memory so as

(i) to receive a request from a remote source at the network server,

(ii) to determine whether to process the request in the network server by executing process steps so as (a) to determine a load currently being processed by the first network server, and (b) to receive information in the first network server from each of the other network servers, the information from each of the other network servers comprising information concerning a load currently being processed in each network server;

wherein the processor determines that the first network server should process the request in a case that (i) the load currently being processed in the first network

server is below a first predetermined level, or (ii) the load currently being processed in the first network server is above the first predetermined level and is above loads currently being processed by either of the other network servers by less than a second predetermined level; and

wherein the processor determines that the first network server should not process the request in a case that the load currently being processed in the first network server is above the first predetermined level and a load currently being processed in at least one of the other network servers is below the level of the first network server by at least the second predetermined level;

(iii) to process the request in the network server in a case that the processor determines that the request should be processed in the network server, and

(iv) to route the request to another one of the plurality of network servers in a case that the processor determines that the request should not be processed in the network server.

6. Computer-executable process steps stored on a computer-readable medium, the computer executable process steps comprising a server module which is installable in a plurality of network servers to distribute requests among the plurality of network servers, the computer-executable process steps comprising:

code to receive a request from a remote source at a first one of the network servers;

code, executable by the first server to determine whether to process the request in that server;

code to process the request in the first network server in a case that the determining code determines that the request should be processed in the first network server; and

code to route the request to another network server in a case that the determining code determines that the request should not be processed in the first network server;

wherein the determining code comprises code to determine whether the request is related to a stateful transaction based on a URI in the request; and

wherein (i) in a case that the request is related to a stateful transaction, the determining code determines that the request should be processed in the first network server, and (ii) in a case that the request is not related to a stateful transaction, the determining code determines if the request should be processed in the first network server.

7. Computer-executable process steps according to claim 6, wherein, in a case that the request is related to a stateful transaction, the code to determine determines that at least a second request having a URI substantially the same as the URI of the request should be processed in the first network server.

First Hit Fwd Refs

Generate Collection

L9: Entry 9 of 24 File: USPT Aug 20, 2002

DOCUMENT-IDENTIFIER: US 6438652 B1
TITLE: Load balancing cooperating cache servers by shifting forwarded request

Application Filing Date (1):
19981009

Brief Summary Text (5):
With cooperating cache servers, a request that cannot be serviced locally due to a cache miss can be forwarded to another cache server storing the requested object. As a result, there are two kinds of requests that can come to a cache server: direct requests and forwarded requests. Direct requests are those that are received directly from clients. Forwarded requests are those that come from other cooperating cache servers on behalf of their clients due to cache misses on the cache servers. With requests forwarded among the cache servers, a cache server can easily become overloaded if it happens to contain in-demand (or "hot") objects that most clients are currently interested in, creating uneven workloads among the cache servers. Uneven workloads can create a performance bottleneck, as many of the cache servers are waiting for the same overloaded cache server to respond to requests forwarded to it. Therefore, there is a need for a way to perform dynamic load balancing among a collection of proxy cache servers. The present invention addresses such a need.

Brief Summary Text (7):
Cooperative caching, or remote caching, has been used in distributed file systems to improve system performance (see "Cooperative caching: Using Remote Client Memory to Improve File System Performance," by M. D. Dahlin et al., Proc. of 1st Symp. on Operating Systems Design and Implementation, pp. 1-14, 1994). Here, the file caches of a collection of workstations distributed on a LAN are coordinated to form a more effective overall file cache. Each workstation caches not only objects referenced by local requests but also objects that may be referenced by requests from a remote workstation. Upon a local cache miss, a local request can be sent to other client workstations where a copy can be obtained, if found. Otherwise, the object is obtained from the object server. The emphasis here is mainly how to maintain cache coherency in the face of updates and how to maintain cache hit ratios by moving a locally replaced object to the cache memory of another workstation. There is no dynamic load balancing.

Brief Summary Text (10):
Yet another way to locate an object on a cooperating cache server is through a hash function. An example is the Cache Array Routing Protocol (CARP) (see V. Valloppillil and K. W. Ross, "Cache Array Routing Protocol v1.0," Internet Draft, http://ircache.nlanr.net/Cache/ICP/draft-vinod-carp-v1-03.txt, February 1998). In CARP, the entire object space is partitioned among the cooperating cache servers, with one partition for each cache server. When a request is received by a cache server from a configured client browser, a hash function is applied to a key from the request, such as the URL or the destination IP address, to identify the partition. If the hash partition is the assigned to requesting cache server, then the request is serviced locally. Otherwise, it is forwarded to the proper cache server in the identified partition.

Detailed Description Text (3):
According to the present invention a logical load monitor 120 includes a load balancing logic 130 for monitoring the load conditions and forwarding frequency (FIG. 2a) of the cooperating cache servers 150 and provides load balancing for them. As will be described below, various load monitor 120 features can: reside in one or more of the cache servers; be duplicated and distributed among the cache servers; or reside in another dedicated system such as a personal computer (PC) server or workstation. In a centralized system configuration, the load monitor 120 can perform a central directory function in directing forwarded requests 125 to the cache servers. One or more browsers 160 can be configured to connect to each cache server 150. Direct requests 155 are sent from the clients such as computers running conventional browsers 160 to the configured cache server 150. If the requested object can be found locally, then it is returned to the browser. Otherwise, the cache server 150 communicates a message to the load monitor 120. Various example implementations of the load monitor 120 will be described in more detail below. If no load imbalance condition or trend exits, the load monitor 120 then forwards the request 125 to the cache server 150 that owns the requested object. The owning cache server then sends the requested object to the requesting cache server, e.g., via the LAN 140.

Detailed Description Text (14):
Here, if a cache server 150 has a cache miss, the local load monitor 120' looks up the ownership of the requested object in its local caching table 101 and forwards the request, to the owning cache server. Alternatively, the hash function could be applied to a key from the request, such as the URL or the destination IP address, to identify the partition and the request then forwarded to the correct cache server. When the forwarded request (i.e., from a cache server who had a cache miss) is received, the owning cache server identifies it as a forwarded request (e.g., by identifying it as from another cache server as opposed to a client) and updates its forwarding frequency 1011 information as applicable (FIG. 3, step 203). If an overload trend or condition is indicated (step 204), the owning cache server can respond to the requesting cache server with a shift request and a copy of the cached object. Alternatively, the requesting cache server can obtain a copy from the originating object server via an intranet, WAN or Internet 110. In either case, when the forwarding server caches a copy of the object, this server will no longer issue forward requests (steps 301, 302) as long as it remains in the cache, thus proportionally reducing the load on the owning server. In addition, the owning cache server can multicast a shift request message to one or more of the other cooperating cache servers 150 so that subsequent forward requests will be shifted, e.g., by updating their local copy of the caching table or modifying the hash function (step 205). At this point, other cache servers can forward their requests to the new owner (or to the least loaded owner of two or more cache servers 150 if ownership is shared) as indicated in their local copy of the caching table 101. When the original cache owner's load has decreased to an acceptable level (step 204), e.g., as indicated by a threshold, the shared ownership information can be merged to its original state (e.g., B,A 10121.fwdarw.B).

CLAIMS:

20. The system of claim 1, further comprising, means for calculating the load condition of each cache server in past intervals; means for computing a mean load of all cache servers in past intervals; and means for finding the cache servers that exceed a threshold above said mean load.

46. The method of claim 28, further comprising the steps of, calculating the load condition of each cache server in past intervals; computing a mean load of all cache servers in past intervals; and finding the cache servers that exceed a threshold above said mean load.

54. The method of claim 53, wherein said step of monitoring the load condition

comprises the steps of, calculating the load condition of each cache server in past intervals; computing a mean load of all cache servers in past intervals; and finding those cache servers that exceed a threshold above said mean load.

81. The program storage device of claim 63, further comprising the steps of, calculating the load condition of each cache server in past intervals; computing a mean load of all cache servers in past intervals; and finding the cache servers that exceed a threshold above said mean load.

89. The program storage device of claim 88, wherein said step of monitoring the load condition comprises the steps of, calculating the load condition of each cache server in past intervals; computing a mean load of all proxy cache servers in past intervals; and finding those proxy cache servers that exceed a threshold above said mean load.

First Hit Fwd Refs

Generate Collection

L9: Entry 11 of 24 File: USPT May 14, 2002

DOCUMENT-IDENTIFIER: US 6389448 B1
TITLE: System and method for load balancing

Abstract Text (1):
A system for distributing load between multiple servers where more than one server in a sever cluster receives a request for connection from a client and each server makes a determination of whether or not to respond to the request. Software modules running on the servers monitor and communicate relative abilities of each server to respond to client requests. Each server responding to a percentage of client requests corresponding to its relative ability to respond.

Application Filing Date (1):
20000505

Brief Summary Text (4):
With the rapid proliferation in use of distributed data networks such as the Internet, more and more clients from around the world are attempting to connect to and extract data stored on a finite number of servers. Those establishing and maintaining the servers containing the desired data, such as web pages from popular web sites, are finding it difficult to insure that all the clients attempting to access data will be able to do so. A given server can only connect with a finite number of clients at the same time. The number of simultaneous connections a given server can support is a function of the server's computational, storage and communications capabilities. In situations where the number of clients attempting to access data stored on a server exceeds the server's capacity, some clients either will not be able to connect or will be dropped by the server. In other cases where a server is overwhelmed by client requests for data, the server may cease to function altogether.

Brief Summary Text (5):
As a partial solution to the situation described above, server operators will typically deploy multiple mirrored servers, each having data identical to that stored on all the other servers. The mirrored servers are typically connected to the same network and are referred to as a server cluster. In conjunction with the multiple mirrored servers, a load balancer is typically used. When a client attempts to connect to and access data from a server cluster, the client's request is first received by the load balancer which determines which of the servers is best suited to handle the client's request. Various load balancing solutions are commercially available and each uses different techniques and criteria for determining to which server to direct a client's request. However, the common characteristic for each of the currently available solutions is that they all attempt to pick a server which is most capable of responding to the client's request.

Brief Summary Text (6):
An inherent drawback with the load balancing solutions available today is that they all require a device generally referred to as a load balancer. As described above, the load balancer is the first point of contact for each client attempting to access data on the server cluster, and therefore, the maximum rate at which the entire server cluster can receive and respond to client requests is limited by the

throughput of the load balancer. It is foreseeable that the number of client requests for data may exceed a load balancer's ability to properly route the requests, and requests will be ignored or dropped despite the fact that the server cluster has sufficient capacity to handle all the request. Another drawback of the currently available load balancers is that when they malfunction, their entire server cluster becomes inoperative.

Brief Summary Text (11):
The servers are connected to the network in parallel such that each server receives every connection request, such as a synchronizing segment or packet (SYN) for transmission control protocol/internet protocol (TCP/IP) connection, substantially at the same time. The SYN packet is the first segment or packet sent by the TCP protocol and is used to synchronize the two ends of a connection in preparation for opening a connection. The load balancing modules on the servers communicate with each other to determine each server's relative ability to accept a new connection (i.e., available capacity).

Detailed Description Text (8):
Although, the SYN packet is transmitted to each server in the cluster, only one server, namely the associated load balancing module 12, accepts and establishes a connection to the client computer 60 based on the connection value of the SYN packet. The load balancing module 12 on each server compares the connection value of a given SYN Packet to its respective assigned sub-range of connection values at step 402. If the connection value of the SYN packet is within the server's assigned sub-range, the associated load balancing module 12 forwards the SYN packet to the TCP stack and the server accepts the connection request from the client computer 60 that transmitted this SYN packet at step 403. However, if the connection value of the SYN packet is not within the server's assigned sub-range, the associated load balancing module 12 will then simply discard the SYN packet at step 404.

Detailed Description Text (11):
In one embodiment of the present invention, each server receives the availability information from each other server and determines its own sub-range. Based on availability information from its own agent 14 and other agents 14 associated with other servers in the cluster, each load balancing module 12 calculates a new sub-range of connection values that it will accept. Alternatively, the coordinating device 20 determines and transmits a new sub-range for each server based on all of the availability information from each server 10. The coordinating device 20 or each server in the cluster continually adjusts the sub-range for each server to reflect the servers' availabilities to accept new connections.

CLAIMS:

1. A system for distributing load within a network, comprising:

at least one cluster having a plurality of servers connected to each other, each server in said cluster being addressable by a common network address; and

a load balancing module in each server for calculating a connection value for each connection request addressed to said network address and for accepting a connection request as a function of available capacity of said respective server with respect to overall available capacity of said cluster and the connection value associated with said connection request.

11. A system for distributing load within a network, comprising:

at least one cluster having a plurality of servers connected to each other, each server in said cluster being addressable by a common network address;

a plurality of routers for routing connection requests to said plurality of

servers, each server in said cluster connected to said at least one router; and

a load balancing module in each server for calculating a connection value for each connection request addressed to said network address and for accepting a connection request from a router as a function of available capacity of said respective server with respect to overall available capacity of servers connected to said router and the connection value associated with said connection request.

First Hit Fwd Refs

Generate Collection

L9: Entry 20 of 24 File: USPT Jun 20, 2000

DOCUMENT-IDENTIFIER: US 6078960 A
TITLE: Client-side load-balancing in client server network

Abstract Text (1):
Load balancing is achieved at the client side, rather than at the server side of a client-server network. Each client computer regularly receives a load balance list, enumerating respective addresses of multiple server computers. Each client computer executes a server selection function which determines the average load for each server in the list. In the event of a server computer failure, a system administrator can remove the server computer from the load balance list and reapportion the load. The client computer's list then is updated when the list is received during subsequent access. In the event a client computer determines that a server is non-responsive, such server is removed from the load balance list for the client computer which made such determination. The client computer also stores a back-up list of servers for use when all servers on the load balance list are non-responsive.

Application Filing Date (1):
19980703

Brief Summary Text (3):
In large client server networks such as the INTERNET, many client computers seek to access the same network server computer at a given time. To do so, each client computer issues an access request which specifies an address for the server computer. When too many clients attempt to access the same server computer at a given time, response time slows as the server handles a queue of requests. To handle the load, it is common for INTERNET service providers to include multiple computers which are linked together to have the same address. For example, a hardware device may respond to all requests to a given address and forward the respective requests to any one of a plurality of redundant network servers. Such hardware device balances the load among the redundant servers allowing response time to be maintained at a desirable speed. Such a hardware device typically handles 16, 64 or 128 servers. To assure that the client is able to access desired data, all accessible data is to be redundantly located on each server computer having the same address. Thus, the multiple servers sharing a common physical address on the client server network are referred to herein as redundant servers.

Brief Summary Text (11):
According to another aspect of the invention, in the event a client computer determines that a server is non-responsive, such server is removed from the local load balance list of the client computer which made such determination. As a result, traffic from such client computer is automatically diverted to other servers that are functioning. Eventually when the client computer reobtains a load balance list from another server, an updated list will be available for the client computer, (assuming the system administrator independently has already determined that such non-responsive server is down).

Detailed Description Text (10):
To access data over the world wide web, for example, a client computer specifies a uniform resource locator (URL), which is an address for the data. Such address

includes the address of a computer which stores the data. The server computers 12 in the client-server network 10 are subject to a varying load depending on the access requests at any given time from the client computers 14. If many client computers 14 desire to access the same data at the same time, then many of the client computers 14 may be delayed. Conventionally, one manner in which such demand is handled is by using caches at other computers. An independent service provider, for example, may keep a copy of popular web pages in their resident cache. Another manner, as described in the background section, is to include a hardware load balancing device which responds to accesses for a given address. Multiple servers are coupled to the hardware load balancing device. A request from a client computer is routed to one of such servers through the load balancing device. In effect the server computers coupled to the load balancing device have the same address. Consider an example in which 10 server computers are coupled to a hardware load balancing device. Such device implements an algorithm to determine how to apportion the load (e.g., the requests to such common address) among the 10 computers. The apportionment may be a simple equal division (e.g., 10%), or may be according to some algorithm (e.g., round robin, random selection, a mathematical formula). These approaches are server side load balancing approaches, meaning the determination is made at the server side of the network.

Detailed Description Text (13):
A given block of data offered over the client-server INTERNET network is controlled by a given person or company. For example, the ABC Company may offer publicly accessible data from their web pages over the INTERNET. Because of popularity of another reason, the ABC Company needs to locate the data on multiple server computers to handle all the demand of client computers seeking to access their web site. The ABC Company stores redundant copies of the web site data on each of multiple servers. According to an aspect of this invention, such servers have differing addresses. Apportionment of the load to determine which server is accessed by which client computer request is determined at the client computers 14.

Detailed Description Text (17):
Referring to FIG. 6, a method for client-side load balancing includes a step 50 in which a client computer 14 requests to access data over the client-server network 10. At another step 52, the client computer 14 executes a server selection function to determine which server 12 to access. A processor 28, for example, reads the load balance list 54 resident on the client computer 14. The server selection function determines which server identified in the list 54 is to be accessed to handle the pending data request. For an embodiment in which there is a load percentage included in the list 54, such percentages are seeds for the server select function. Over time as the server select function is executed over and over, the actual load percentage for each server computer in the list 54 converges to the specified percentage in the list 54. According to an alternative scheme, the load select function may randomly select one of the servers in the list 54 or perform a round-robin selection, or perform some mathematical computation.

Detailed Description Text (20):
In some instances, the attempt to connect to a server at step 56 may fail. This may occur because the server is not longer a participating server, or because the server is down, or for some other reason. If the connection and access is unsuccessful, then at step 64 a response analysis is performed. The response analysis determines at step 66 whether the server should be deemed non-responsive and thus removed from the load balance list 54 at step 68, or whether to simply try another server. In one embodiment a server is deemed non-responsive if the data is not present at the server or if the connection fails after a prescribed number of tries. If the response analysis results in the server being found non-responsive then the server is removed from the load balance list of the local client computer which made the determination. Whether responsive or non-responsive, the list 54 then is checked at step 70 to determine if there another server 12 on the load

bálance list. If yes, then at step 72 connection to another server in the list 54 is attempted. The operation then continues with the test at step 58. If the check at step 70 results in a determination that no others servers are identified in the list 54, then at step 74 a server 15 from the emergency back-up load balance list 76 is accessed. The client computer 14 then accesses the data directly from the identified server 15, or merely receives an updated load balance list 54 and tries again.

Detailed Description Text (21):
Note that a client computer may include multiple load balance lists. One load balance list may be used for determining access to one set of data (e.g., ABC Company data) over the client-server network 10, while another load balance list is used for determining access to another set of data. The servers in each list may be the same or differ.

CLAIMS:

1. A method of apportioning load in a client server network having a first plurality of server computers, wherein load is apportioned among a second plurality of the server computers, the second plurality of server computers being a subset of the first plurality of server computers, the method comprising the steps of:

storing a load balance list on each server computer of the second plurality of server computers;

storing common data on each server computer of the second plurality of server computers;

for each one of a plurality of client computers connecting to the client server network, receiving the load balance list from a server computer of the second plurality of server computers;

performing a server selection function at a given client computer among the plurality of client computers using data from the received load balance list to identify a select server computer among the second plurality of server computers;

responsive to the performing step, accessing the common data from the select server computer with the given client computer; and

updating the load balance list for the client server network, the step of updating comprising the steps of:

(i) storing an updated load balance list on each one of a third plurality of server computers among the first plurality of server computers, wherein the updated load balance list identifies each server computer of the third plurality of server computers; and

(ii) receiving at one or more of the plurality of client computers the updated load balance list from a server computer of the third plurality of server computers;

wherein the step of performing the server selection function at a given client computer among the plurality of client computers uses data from the received updated load balance list to identify a select server computer among the third plurality of server computers.

4. A method of apportioning load in a client server network having a first plurality of server computers, wherein load is apportioned among a second plurality of the server computers, the second plurality of server computers being a subset of the first plurality of server computers, the method comprising the steps of:

storing a load balance list on each server computer of the second plurality of server computers;

storing common data on each server computer of the second plurality of server computers;

for each one of a plurality of client computers connecting to the client server network, receiving the load balance list from a server computer of the second plurality of server computers;

performing a server selection function at a given client computer among the plurality of client computers using data from the received load balance list to identify a select server computer among the second plurality of server computers; and

responsive to the performing step, accessing the common data from the select server computer with the given client computer;

wherein the step of storing a load balance list comprises the step of storing a first load balance list on at least one server computer among the second plurality of server computers, and storing a second load balance list different from the first load balance list on at least one other server computer among the second plurality of server computers.

5. A method of apportioning load in a client server network having a first plurality of server computers, wherein load is apportioned among a second plurality of the server computers, the second plurality of server computers being a subset of the first plurality of server computers, the method comprising the steps of:

storing a load balance list on each server computer of the second plurality of server computers;

storing common data on each server computer of the second plurality of server computers;

for each one of a plurality of client computers connecting to the client server network, receiving the load balance list from a server computer of the second plurality of server computers;

performing a server selection function at a given client computer among the plurality of client computers using data from the received load balance list to identify a select server computer among the second plurality of server computers; and

responsive to the performing step, accessing the common data from the select server computer with the given client computer;

wherein the load balance list is a first load balance list, and further comprising the steps of:

(i) storing a back-up load balance list on a plurality of client computers, wherein the back-up load balance list identifies at least one server computer not among the second plurality of server computers;

(ii) determining by one of the plurality of client computers which stores the back-up load balancing list that all server computers identified in the first load balance list are non-responsive; and

(iii) connecting said one of the plurality of client computers to a server computer identified in the back-up load balance list.

7. A client server network, comprising:

a plurality of server computers;

a plurality of client computers;

a load balance list identifying a subset of the plurality of server computers, wherein the load balance list is stored at one of the plurality of client computers, wherein the load balance list is a first load balance list,

common data stored at each one server computer of the server computers identified in the first load balance list;

processing means at said one of the plurality of client computers which executes a selection function to select one server computer of the server computers identified in the first load balance list, wherein the selection function apportions load among the server computers identified in the first load balance list;

means for connecting said one of the plurality of client computers to the selected one server computer to access a portion of the common data;

a back-up load balance list stored at said one of the plurality of client computers, wherein the back-up load balance list identifies at least one server computer not identified in the first load balance list;

means for determining at said one of the plurality of client computers that all server computers identified in the first load balance list are non-responsive; and

means for connecting said one of the plurality of client computers to a server computer identified in the back-up load balance list.

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, April 21, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L15	5721904[uref]	14
☐	L14	20001101	7
☐	L13	(call-up or inquire) near5 load near5 (information or data)	8
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L12	(call-up or inquire) near5 load near5 (information or data)	4
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L11	client-based adj3 (balancing or balanced)	2
☐	L10	client-based near8 (balancing or balanced)	3
☐	L9	l8 and l6	24
☐	L8	server near8 (determine or determining or calculate or calculating or identify or identifying) near8 (load or loading)	1009
☐	L7	l6 and (RDP or ICA)	1
☐	L6	l2 and ((user or computer or terminal or client) near8 (request or inquiery))	124
☐	L5	20001101	6
☐	L4	(load-balanced adj4 access)	8
☐	L3	(load-balanced adj4 access) same ((user or computer or terminal) near8 (request or inquiery))	0
☐	L2	20001101	124
☐	L1	(load-balancing or load-balanced or(load near4 balancing) or (load-balanced adj4 access)) same ((user or computer or terminal) near8 (request or inquiery))	404

END OF SEARCH HISTORY

First Hit

☐ Generate Collection

L9: Entry 1 of 24 File: PGPB Feb 28, 2002

DOCUMENT-IDENTIFIER: US 20020026560 A1
TITLE: LOAD BALANCING COOPERATING CACHE SERVERS BY SHIFTING FORWARDED REQUEST

Application Filing Date:
19981009

Summary of Invention Paragraph:
[0003] With cooperating cache servers, a request that cannot be serviced locally due to a cache miss can be forwarded to another cache server storing the requested object. As a result, there are two kinds of requests that can come to a cache server: direct requests and forwarded requests. Direct requests are those that are received directly from clients. Forwarded requests are those that come from other cooperating cache servers on behalf of their clients due to cache misses on the cache servers. With requests forwarded among the cache servers, a cache server can easily become overloaded if it happens to contain in-demand (or"hot") objects that most clients are currently interested in, creating uneven workloads among the cache servers. Uneven workloads can create a performance bottleneck, as many of the cache servers are waiting for the same overloaded cache server to respond to requests forwarded to it. Therefore, there is a need for a way to perform dynamic load balancing among a collection of proxy cache servers. The present invention addresses such a need.

Summary of Invention Paragraph:
[0005] Cooperative caching, or remote caching, has been used in distributed file systems to improve system performance (see"Cooperative caching: Using Remote Client Memory to Improve File System Performance," by M. D. Dahlin et al., Proc. of 1st Symp. on Operating Systems Design and Implementation, pp. 1-14, 1994). Here, the file caches of a collection of workstations distributed on a LAN are coordinated to form a more effective overall file cache. Each workstation caches not only objects referenced by local requests but also objects that may be referenced by requests from a remote workstation. Upon a local cache miss, a local request can be sent to other client workstations where a copy can be obtained, if found. Otherwise, the object is obtained from the object server. The emphasis here is mainly how to maintain cache coherency in the face of updates and how to maintain cache hit ratios by moving a locally replaced object to the cache memory of another workstation. There is no dynamic load balancing.

Summary of Invention Paragraph:
[0008] Yet another way to locate an object on a cooperating cache server is through a hash function. An example is the Cache Array Routing Protocol (CARP) (see V. Valloppillil and K. W. Ross, "Cache Array Routing Protocol v 1.0," Internet Draft, http://ircache.nlanr.net/Cache/ICP/draft- -vinod-carp-v1-03.txt, February 1998). In CARP, the entire object space is partitioned among the cooperating cache servers, with one partition for each cache server. When a request is received by a cache server from a configured client browser, a hash function is applied to a key from the request, such as the URL or the destination IP address, to identify the partition. If the hash partition is the assigned to requesting cache server, then the request is serviced locally. Otherwise, it is forwarded to the proper cache server in the identified partition.

Detail Description Paragraph:
[0027] According to the present invention a logical load monitor 120 includes a load balancing logic 130 for monitoring the load conditions and forwarding frequency (FIG. 2a) of the cooperating cache servers 150 and provides load balancing for them. As will be described below, various load monitor 120 features can: reside in one or more of the cache servers; be duplicated and distributed among the cache servers; or reside in another dedicated system such as a personal computer (PC) server or workstation. In a centralized system configuration, the load monitor 120 can perform a central directory function in directing forwarded requests 125 to the cache servers. One or more browsers 160 can be configured to connect to each cache server 150. Direct requests 155 are sent from the clients such as computers running conventional browsers 160 to the configured cache server 150. If the requested object can be found locally, then it is returned to the browser. Otherwise, the cache server 150 communicates a message to the load monitor 120. Various example implementations of the load monitor 120 will be described in more detail below. If no load imbalance condition or trend exits, the load monitor 120 then forwards the request 125 to the cache server 150 that owns the requested object. The owning cache server then sends the requested object to the requesting cache server, e.g., via the LAN 140.

Detail Description Paragraph:
[0038] Here, if a cache server 150 has a cache miss, the local load monitor 120' looks up the ownership of the requested object in its local caching table 101 and forwards the request, to the owning cache server. Alternatively, the hash function could be applied to a key from the request, such as the URL or the destination IP address, to identify the partition and the request then forwarded to the correct cache server. When the forwarded request (i.e., from a cache server who had a cache miss) is received, the owning cache server identifies it as a forwarded request (e.g., by identifying it as from another cache server as opposed to a client) and updates its forwarding frequency 1011 information as applicable (FIG. 3, step 203). If an overload trend or condition is indicated (step 204), the owning cache server can respond to the requesting cache server with a shift request and a copy of the cached object. Alternatively, the requesting cache server can obtain a copy from the originating object server via an intranet, WAN or Internet 110. In either case, when the forwarding server caches a copy of the object, this server will no longer issue forward requests (steps 301, 302) as long as it remains in the cache, thus proportionally reducing the load on the owning server. In addition, the owning cache server can multicast a shift request message to one or more of the other cooperating cache servers 150 so that subsequent forward requests will be shifted, e.g., by updating their local copy of the caching table or modifying the hash function (step 205). At this point, other cache servers can forward their requests to the new owner (or to the least loaded owner of two or more cache servers 150 if ownership is shared) as indicated in their local copy of the caching table 101. When the original cache owner's load has decreased to an acceptable level (step 204), e.g., as indicated by a threshold, the shared ownership information can be merged to its original state (e.g., B,A 10121-->B).

CLAIMS:

19. The method of claim 1, further comprising the steps of, calculating the load condition of each cache server in past intervals; computing a mean load of all cache servers in past intervals; and finding the cache servers that exceed a threshold above said mean load.

27. The method of claim 26, wherein said step of monitoring the load condition comprises the steps of, calculating the load condition of each cache server in past intervals; computing a mean load of all proxy cache servers in past intervals; and finding those proxy cache servers that exceed a threshold above said mean load.

54. The program storage device of claim 36, further comprising the steps of,

calculating the load condition of each cache server in past intervals; computing a mean load of all cache servers in past intervals; and finding the cache servers that exceed a threshold above said mean load.

62. The program storage device of claim 61, wherein said step of monitoring the load condition comprises the steps of, calculating the load condition of each cache server in past intervals; computing a mean load of all proxy cache servers in past intervals; and finding those proxy cache servers that exceed a threshold above said mean load.

90. The system of claim 71, further comprising, means for calculating the load condition of each cache server in past intervals; means for computing a mean load of all cache servers in past intervals; and means for finding the cache servers that exceed a threshold above said mean load.

First Hit Fwd Refs

Generate Collection

L9: Entry 2 of 24 File: USPT Apr 20, 2004

DOCUMENT-IDENTIFIER: US 6725253 B1
TITLE: Load balancing system

Abstract Text (1):
The load balancing system comprises a first load balancing apparatus (primary), a first load balancing apparatus (secondary), and a first load balancing apparatus (backup), which carry out load balancing by primary-sorting service requests from clients to service sites; and first and second secondary load balancing apparatuses which carry out load balancing by second-sorting the service request which has been primary-sorted by the first load balancing apparatus (primary), the first load balancing apparatus (secondary) and the first load balancing apparatus (backup), to one of multiple servers provided in the site.

Application Filing Date (1):
20000728

Brief Summary Text (16):
The server 6 is provided on the service supplier side and connects to the LAN/WAN 4. The server 6 comprises a computer terminal for supplying service to the client 2 in accordance with a service request therefrom. The server 7 is similarly provided on the service supplier side as the server 6 and connects to the LAN/WAN 4. The server 7 comprises a computer terminal for supplying service to the client 2 in accordance with a service request therefrom. The server 6 and the server 7 are provided in physically different places. Therefore, the distances between the servers 6 and 7 and the client 2 (the client-side DNS server 3) are different.

Brief Summary Text (20):
The server-side DNS server 5 is responsible for the servers 6 and 7, and when an inquiry to convert a representative domain name to an IP address is received from the client-side DNS server 3, the server-side DNS server 5 responds to the client-side DNS server 3 by converting the representative domain name to the IP address of either one of the servers 6 and 7. That is, the server-side DNS server 5 sorts the access request from the client 2 by allocating it to either one of the servers 6 and 7. This sorting constitutes the load balancing mentioned above.

Brief Summary Text (41):
The client 11 comprises a computer end terminal for requesting service from a server 13 (and a server 14) explained later via the LAN/WAN 10. A load balancing apparatus 12 balances the load between the servers 13 and 14 by sorting access requests from the client 11 to either one of the servers 13 and 14 by means of NAT and MAC as described above. A representative IP address represents the servers 13 and 14 and is appended to the load balancing apparatus 12.

Brief Summary Text (42):
The servers 13 and 14 are provided on the service supplier side and connect to the LAN/WAN 10. The servers 13 and 14 comprise computer terminals for supplying service to the client 11 in accordance with service requests therefrom. By using the above-mentioned representative IP address, the servers 13 and 14 function as a single server from the point of view of the client 11. An IP address is appended to each of the servers 13 and 14. That is, the servers 13 and 14 form one virtual server

which is expressed by the representative IP address, and the client 11 accesses the representative IP address of this virtual server.

Brief Summary Text (43):
To access the virtual server in the above constitution, for example in a step SC1 the client 11 accesses the load balancing apparatus 12 having the representative IP address via the LAN/WAN 10. In a step SC2, the client 12 sorts an IP packet (service request) from the client 11 and allocates it to the server 14, for example, by means of NAT and MAC.

Brief Summary Text (52):
In order to achieve the above objects, a first aspect of this invention provides a load balancing system applied in a network system comprising a plurality of clients (corresponding to the clients 100 and 200 in the embodiment described later), and a plurality of servers (corresponding to the servers 510, 520, 610 and 620 in the embodiment described later) which are separately provided in a plurality of sites (corresponding to the service sites SS.sub.1 and SS.sub.2 in the embodiment described later) and supply services to the plurality of clients, the load balancing system balancing service requests from the plurality of clients, and comprising a plurality of first load balancing units (corresponding to first load balancing apparatuses 410, 420 and 430 in the embodiment described later) which are provided physically separate and balance load by first-sorting service requests from the plurality of clients to the plurality of sites; and a plurality of second load balancing units (corresponding to a second load balancing apparatus 500 and a second load balancing apparatus 600 in the embodiment described later) which are provided in correspondence with the plurality of sites and each carry out load balancing by second-sorting a service request which has been first-sorted by the first load balancing unit to one of a plurality of servers provided in the site; one of the plurality of first load balancing units being set as a primary, the primary first load balancing unit monitoring the operating status of the other first load balancing units and the plurality of second load balancing units, and removing a unit which has stopped operating from the units which carry out load balancing.

Brief Summary Text (53):
According to the invention of the first aspect, when a client makes a request for service, one of the plurality of first load balancing units first-sorts the service request to one site (a second load balancing unit). Consequently, the second load balancing unit second-sorts the service request to one server.

Brief Summary Text (54):
In the case where one of multiple first load balancing units has stopped operating as a result of a disaster, this first load balancing unit is excluded from the units which balance the load by first-sorting. Therefore, in this case, the excluded first load balancing unit does not carry out first-sorting even when a client sends a service request. Instead, the service request is first-sorted by the other first load balancing unit which is operating normally.

Detailed Description Text (10):
The first load balancing apparatus group 400 transmits the IP address of either one of the second load balancing apparatus 500 and the second load balancing apparatus 600 in response to a DNS inquiry from the client-side DNS server 110 and the client-side DNS server 210, based on a sorting table and the like described later. That is, the first load balancing apparatus group 400 executes first load balancing by carrying out first-sorting of access requests from the client 100 and the client 200 into those destined for the second load balancing apparatus 500 and those destined for the second load balancing apparatus 600.

Detailed Description Text (14):
Triggered by a measurement request from the first load balancing apparatus group

400, the second load balancing apparatus 500 measures the load in the communications path to, for example, the client-side DNS server 110 (server 510 and server 520). The server 510 is provided on the service supplier side, and comprises a computer terminal for supplying service to the client 100 in accordance with a service request based on an access by the client 100. Similarly, the server 520 is provided on the service supplier side, and comprises a computer terminal for supplying service to the client 100 in accordance with a service request based on an access by the client 100.

Detailed Description Text (16):
In the same way as the second load balancing apparatus 500, the second load balancing apparatus 600 is triggered by a measurement request from the first load balancing apparatus group 400 to measure the load in the communications path to, for example, the client-side DNS server 210 (server 610 and server 620). The server 610 is provided on the service supplier side, and comprises a computer terminal for supplying service to the client 200 in accordance with a service request based on an access by the client 200. Similarly, the server 620 is provided on the service supplier side, and comprises a computer terminal for supplying service to the client 200 in accordance with a service request based on an access by the client 200.

Detailed Description Text (92):
Similarly, in a step SH3, the first load balancing apparatus (primary) 410 sends a request to measure the path load to the second load balancing apparatus 600. Consequently, in a step SH4, the second load balancing apparatus 600 determines the effective bandwidth of the communications path to the client-side DNS server 110 by using the same method as that used by the second load balancing apparatus 500.

Detailed Description Text (95):
Subsequently, an access process in this embodiment will be explained with reference to FIGS. 16 and 17. In FIG. 16, in a step SI1 the client 100 sends a DNS inquiry based on a representative domain name to the client-side DNS server 110. The client-side DNS server 110 determines whether the IP address of the second load balancing apparatus 500 or the second load balancing apparatus 600 which is the response is stored in the cash memory of the client-side DNS server 110. In the case where the IP address is stored, in a step SI4 the IP address is sent as the response to the client 100.

Detailed Description Text (99):
Consequently, in a step SJ1 of FIG. 17, the client 100 accesses the second load balancing apparatus 600 based on the above-mentioned IP address. In a step SJ2, the second load balancing apparatus 600 sorts the IP packet (service request) received from the client 100 to, for example, the server 620 by using NAT and MAC. In a step SJ3, the server 620 transmits a reply packet for supplying service via the second load balancing apparatus 600 or directly to the client 100. As a consequence, the service is supplied from the server 620 to the client 100.

CLAIMS:

1. A load balancing system applied in a network system comprising a plurality of clients, and a plurality of servers which are separately provided in a plurality of sites and supply services to said plurality of clients, the load balancing system balancing service requests from said plurality of clients, and comprising: a plurality of first load balancing units which are provided physically separate and balance load by primary-sorting service requests from said plurality of clients to said plurality of sites; a plurality of second load balancing units which are provided in correspondence with said plurality of sites and each carry out load balancing by secondary-sorting a service request which has been primary-sorted by said first load balancing unit to one of a plurality of servers provided in said site; one of said plurality of first load balancing units being set as a primary,

the primary first load balancing unit monitoring the operating status of the other first load balancing units and said plurality of second load balancing units, and removing a unit which has stopped operating from the units which carry out load balancing.

First Hit Fwd Refs

☐ Generate Collection

L9: Entry 4 of 24 File: USPT Dec 30, 2003

DOCUMENT-IDENTIFIER: US 6671259 B1
TITLE: Method and system for wide area network load balancing

Abstract Text (1):
A system and method of load balancing a network having a plurality of client systems and servers. The load balancing system and method distributes requests from the client systems to different servers on a wide area network by selecting the most optimal server for a specific client request. The client request for a server is received from a client system and the destination point of the client request is resolved in part by a load balancing server selector. The client request is sent to a load balancing server based on the resolved destination point and one server out of a subset of the plurality of servers is selected based on a predetermined criteria, such as network traffic congestion or server load, and a conduit to the client system and the selected server for transmission of the client request is provided.

Application Filing Date (1):
19990330

Brief Summary Text (2):
The present invention relates generally to network load balancing systems and methods, and in particular to methods and systems for selecting one server among many servers in a network by using network measurements and a request by a client system.

Brief Summary Text (5):
Interconnecting the client systems and the servers are a series of cables, such as twisted-pair wire, coaxial cables, or fiber optic cables, and network devices, such as routers, switches, and repeaters. Conventionally, when a client system wants a task to be performed, it seeks a server to perform the task. Through the cables and network devices, the client system conveys its request to a server. If the server accepts the task, the server performs the task, and thereby transfers information to and from the client system. A server, however, does not have to be located in the same LAN as the client system.

Brief Summary Text (7):
However, in a WAN, the total number of client systems easily outnumbers the total number of servers. Therefore, a server can easily get bombarded by a tremendous number of client requests and can thus get overwhelmed. Thus, many times numerous distinct servers are utilized in a manner such that any one of the distinct servers may respond to any specific request. Thus, many parallel servers may be available to respond to client requests, and these servers are sometimes allocated in a round robin fashion to process client requests. However, requests may be of varying nature, both in terms of required communication bandwidth and required extent of processing. Accordingly, specific servers may become overwhelmed.

Brief Summary Text (9):
In order to reduce or regulate this server congestion, the present invention provides a load balancing system and method which distributes the client request to different servers on the WAN by selecting the most optimal server for a specific

client request. The load balancing system causes no or minimal disruption to the client systems, servers and network devices. Since in a WAN the client systems, servers and network devices can number in the thousands or millions, minimal or no effect on the client systems, servers and network devices is preferred.

Brief Summary Text (11):
According to the present invention, load balancing for a network is provided. The network has a plurality of client systems and servers. A client request for a server from a client system is received. A destination point of the client request is resolved and the client request is sent to a load balancing server based on the resolved destination point. One server out of a subset of the plurality of servers is selected based on a predetermined criteria. A conduit to the client system and the selected server for transmission of the client request is provided.

Brief Summary Text (12):
In one embodiment, load balancing system for a network is provided. The network has a plurality of client systems and a plurality of servers. At least one load balancing server adapted to be coupled with the network and at least one load balancing server selector coupled with the at least one load balancing server is also provided. The load balancing server selector is adapted to receive requests from the client systems and to forward the requests to the at least one load balancing server. The at least one load balancing server selects one server out of a subset of the plurality of servers based on a predetermined criteria and the requests from the client systems.

Detailed Description Text (5):
The client systems are, as previously mentioned, also connected to the LBS selector 15 and the two LB servers 17a,b. As is discussed more fully below, the LB server balances network traffic and load amongst the servers of a group of servers. The LBS selector, on the other hand, determines which LB server is responsible for balancing loads for which groups of servers. Thus, requests to communicate with servers by the client system are, generally speaking, referred to the LBS selector, which in turn refers the request to a specific LB server. The LB server then arranges for communications between one of the servers and the client system. Depending on various factors, such as the load on the LB server, the LB server either allows the client system to directly contact a specific server or provides specific server information on a per session basis.

Detailed Description Text (6):
More concretely, the client systems 11a,b send requests to the LBS selector 15. One example of a request sent by a client system is a request for a server on the network to perform a specific task. One such specific task is a request for domain name resolution, which is a request for resolving a logical name of a system, e.g. a server, to a unique series of numbers. This unique series of numbers is an Internet Protocol (IP) number. The IP number is the address of a particular server, and allows messages, sent in packets, to be routed from one system to another in the network.

Detailed Description Text (7):
A request by a client system for domain name resolution is routed to the LBS selector 15. The LBS selector resolves, or interprets, the request from the client system to determine the appropriate LB server for the request. The LB server then selects one of the servers amongst a group of servers to receive the client request. In turn, the selected server performs the task required by the client system.

Detailed Description Text (8):
The LB server selects the selected server so as to balance tasks among the group of servers. In one embodiment of the present invention, the server selected by the LB server is based on network measurements gathered and the request originally sent by

the client system. The network measurements are performed on each of the servers or gathered from each server using network measurement devices and techniques. Such devices and techniques are known in the art. In the present invention, the network measurements includes the measurement of load or network traffic experienced by a server. Load refers to the total amount of client requests which the server is servicing or the total amount of operations being performed by a server. Network traffic is the total amount of data packets (traffic on the network) being carried to each of the servers from the client systems, as well as, from each of the servers to the client systems. Other similar network measurements are gathered or determined in the present invention. By examining the network load measurements, the LB server is able to select the most optimal server for the client request.

Detailed Description Text (9):
Thus, in one embodiment of the LB Server of the present invention, the LB Server characterizes or categorizes each of the servers based on the network load measurements. For instance, if server 18a is experiencing hundreds of client requests within a given time period, server 18a is categorized as a high load server. Conversely, if server 18a is experiencing ten or twenty client requests within a given time period, server 18a is categorized as a low load server. Therefore, the servers based on the network measurements gathered can be categorized or ranked in order, from a low load server to a high load server. Similarly, using network traffic as the network measurement, i.e., measuring the total amount of bytes of information transferred to and from a server, each of the servers are categorized. Hence, the servers connected to the LB servers are ranked in order, from a low network traffic server to a high network traffic server.

Detailed Description Text (10):
Using the categorization or ranking of the servers, the LB servers 17a,b selects an appropriate server from the group of servers 19a and 19b. In one embodiment of the LB server, the LB server selects the lowest load server to handle the client requests. For instance, if the LB server 17a received the client request from the LBS selector 15 and the lowest load server was server 18a, the LB server 17a would select the server 18a to handle the client request.

Detailed Description Text (11):
FIG. 2 illustrates a flow diagram of an operational overview of the abstract model illustrated in FIG. 1. In step 21, a client system requests a specific server address. In step 23, an LBS selector resolves the client request to determine the destination point, i.e., server, that is requested by the client system, and routes the client request to a specific LB server. In step 25, the LB server receives and resolves the client request and the LB server selects a specific server out of a server group based on a predetermined criteria. In step 27, the process provides a location (address) of the specific server selected and provides the address to the client system which initiated the client request.

Detailed Description Text (12):
FIG. 3 illustrates a block diagram of one embodiment of the load balancing system in a typical network. For the sake of clarity of description, a single client system 43, a single LBS selector 65, a single LB server 67, and two servers 69 and 71 are depicted in FIG. 3. The total number of client systems, LBS selectors, LB servers and servers, however, may vary according to different embodiments of the invention. As illustrated in FIG. 3, a client system 43 is coupled with a client DNS system 45. The client system sends client requests, e.g. domain name resolution requests, that is received by the client DNS system. The client DNS system, using a domain naming system/service (DNS), resolves the client request.

Detailed Description Text (15):
Referring back to FIG. 3, the client DNS system 45 using DNS, i.e., the domain resolution hierarchy described above, determines that the client request cannot be resolved by the client DNS. Therefore, the client DNS 45 attempts to consult with

the server DNS system 63, i.e., the client DNS 45 forwards the client request to the server DNS 63. The server DNS 63 resolves the client request by providing the IP number of the LBS selector 65. The client DNS 45 in response to the IP number sent by the server DNS 63 forwards the client request to the LBS selector 65. The LBS selector 65 examines the client request sent by the client DNS 45 to select a LB server and to forward the client request to that LB server. In one embodiment, the LBS selector 65 resolves the client request and provides the IP number of the LB server 67 to the client DNS 45. The client DNS 45 sends the client request to the LB server 67 in response to the IP number provided by the LBS selector 65. The LB server 67 examines the client request and the network measurements provided for the servers 69 and 71.

Detailed Description Text (16):
Based on the network measurements and the client request, the LB server 67 determines the best server for handling the client request. Once the determination is made by the LB server 67 and a server is selected, the LB server provides an IP number to the client system 43. Using the provided IP number, the client system directs its request to that IP number. In one embodiment of the LB server using macro-control mode, the IP number provided is a destination point or location of the server on the network. The client system directs its requests directly to the chosen server. In another embodiment of the LB Server using micro-control mode, the IP number provided is a conduit or path. In other words, the LB server opens a conduit or provides a path from the client system to the selected server and the LB server acts as a gatekeeper. Micro-control and macro-control mode is presented in more detail later in the context of FIG. 6.

Detailed Description Text (17):
FIG. 4 illustrates a flow diagram of an operational overview of one embodiment of the load balancing system illustrated in FIG. 3. The client system sends a request for server resolution, i.e., a domain name resolution request, in step 141. In step 143, server resolution occurs such that one or more DNS systems resolves the domain name requested by the client system. Once server resolution is completed, in step 145, the LB server is dynamically configured to work in macro or micro-control mode. If the LB server is dynamically configured to work in macro-control mode, then in step 149, the selected server IP number is provided to the client system and the process ends. If the LB server is dynamically configured in micro-control mode then in step 147, the LB server provides an IP number of the LB server to the client system for a path from the client system to a server through the LB server. The LB server acts as a gatekeeper between the client system and the selected server. Once an IP address is provided to the client system, the process ends.

Detailed Description Text (18):
FIG. 5 illustrates a flow diagram of the step 143 illustrated in the flow diagram in FIG. 4. In step 241, the client system sends a request for server resolution and the request, is examined by a DNS server. If the DNS server cannot resolve the request, the DNS server passes the client request to another DNS server until a server DNS receives the client request. In step 243, the server DNS resolves the client request by identifying an LBS selector as the authoritative name server. An authoritative name server is a server capable of resolving a domain name into an IP number without consulting another server. In step 245, the LBS selector resolves the client request and identifies the LB server as the authoritative name server.

Detailed Description Text (20):
In step 247, the LB server is dynamically configured in macro-control mode or micro-control mode. If the LB server is dynamically configured to be in micro-control mode, than in step 249, the LB server provides an IP number of the LB server such that the LB server acts as a gatekeeper between the client system and a server. In step 251, the LB server selects a server for the client system for a given session and then the process ends. A session includes either a predetermined time period such as five minutes, or a time period measured from the start of the

client request to the completion of a specific task by the server as requested by the client system. If the LB server is dynamically configured to operate in macro-control mode, the LB server selects a server from a group of servers based on a predetermined criteria in step 253. In step 255, the LB server provides the selected server's IP number to the client system which originally initiated the client request and then the process ends.

Detailed Description Text (21):
FIG. 6 illustrates a flow diagram of another embodiment of the load balancing system performing step 143 illustrated in FIG. 4. In step 261, DNS systems passes the client request to other DNS systems until the server DNS receives the client request. In step 263, the server DNS resolves the client request by identifying an LBS selector as a potential authoritative name server. In step 265, the LBS selector resolves the client request to identify the LB server as a source router. The LB server as a source router, selects one IP number out of a set of IP numbers allocated to the LB server for connecting the client system to a specific server.

Detailed Description Text (22):
In other words, instead of the LBS selector responding to the client request directly to the client DNS, the LBS selector directs the LB server to handle the request. Therefore, the LBS selector refrains from sending a response to a client DNS, since the LB server is handling the response. Also, the LB server is dynamically configured to receive the LBS selector requests to handle the client request.

Detailed Description Text (23):
Alternatively, the LB server is dynamically configured to intercept the LBS selector responses intended for a client DNS. The LBS selector responses are then converted into an IP number either representing a path from the client system to a server with the LB server acting as the gatekeeper or a destination point of the server on the network. For instance, a client request is sent to an LBS selector R1. The LBS selector R1 provides a LBS selector response including the IP number of the LB server I1. In other words, the LBS selector sends a LBS selector response to the LB server I1 to force a response, i.e., a route, from the LB server and back to the client DNS. Hence, the LB server I1 receives the LBS selector response and LB server I1, reconfigures the LBS selector response to include the IP number of a path or destination point to a specific server on the network.

Detailed Description Text (25):
In step 265 of the flow diagram illustrated in FIG. 6, the LB server is dynamically configured to operate in micro-control mode or a macro-control mode. Micro-control mode places the LB server in greater control of the flow of information from the client systems to the servers. If the LB server is dynamically configured to be in micro-control mode, the server monitors the information flow on a packet per packet basis and determines the appropriate server to service each packet in step 313. Therefore, the LB server can dynamically change from one server to another quickly and during the same session or connection between the client system and the server. For instance, an LB server initially determines that client system C1 is to be connected to server S1, since server S1 is determined as a low load server and server S2 is determined to be a high load server. During the course of the connection between the client system C1 and server S1, server S1 becomes heavily loaded and server S2 becomes less loaded. Therefore, the server S1 now becomes the high load server and server S2 becomes the low load server. The LB server switches the connection between the client system C1 to the new server S2 on the next client request from the client system. Therefore, the client system is using the "best" server among the two servers based on the conditions at the time a client request is received, with or without knowledge by the client system.

Detailed Description Text (26):
On the other hand, macro-control mode places the LB server in less control of the

flow of information from the client systems to the servers. However, this reduces the overhead experienced by the LB server in having to constantly monitor the information flow on a packet per packet basis and determines the appropriate server to service each packet. If the LB server is dynamically configured to operate in macro-control mode, the LB server selects and determines the client system connection with a server prior to establishing any connection between the client system and the server. For instance, LB server initially determines that client system C1 is to be connected to server S1, since server S1 is determined as a low load server and server S2 is determined to be a high load server. During the course of the connection between the client system C1 and server S1, server S1 becomes heavily loaded and server S2 becomes less loaded. Therefore, the server S1 now becomes the high load server and server S2 becomes the low load server. The LB server does not switch the connection between the client system C1 to the new server S2 on the next packet or client request from the client system.

Detailed Description Text (27):
In another embodiment of the LB server, the LB server in micro-control mode is dynamically configured to switch connection between a client system and a server, if so warranted, after a specified period of time has elapsed. For instance, the LB server initially determines that client system C2 is to be connected to server S2, since server S2 is determined as a low load server and server S3 is determined to be a high load server. After a predetermined time period, such as 5 minutes, the LB server examines the network measurements of the servers. If during the course of the connection between the client system C2 and server S2, server S2 becomes heavily loaded and server S3 becomes less loaded. Therefore, the server S2 now becomes the high load server and server S3 becomes the low load server. The LB server switches the connection between the client system C2 to the server S3 on the next packet or client request from the client system.

Detailed Description Text (28):
In another embodiment of the LB server, the LB server is dynamically configured to switch between micro-control mode and macro-control mode based on the LB server's own network measurements. For instance, the LB server initially is receiving a few client requests, e.g. ten or twenty client requests and thereby switches into micro-control mode. After a predetermined time period, such as 5 minutes, or through continues monitoring, the LB server examines the amount of client requests the LB server is receiving. If LB server determines that the LB server is now heavily loaded, e.g. receiving hundreds of client requests, the LB server switches into macro-control mode. Therefore, the LB server dynamically switches from micro-control mode to macro-control mode and vice versa based on the LB server's own network measurements. In other words, the LB server is able to switch into micro-control mode from macro-control mode, and thereby being in greater control of the flow of information from the client systems to the servers, if the LB server is not heavily loaded. Likewise, the LB server is able to switch into macro-control mode from micro-control mode, and thereby being in less control of the flow of information from the client systems to the servers, if the LB server is heavily loaded.

Detailed Description Text (29):
FIG. 7 illustrates a block diagram of one embodiment of the LBS selector of the present invention. In one embodiment, the LBS selector is a standalone network device or a computing device, e.g. a personal computer, programmed with firmware or software. The LBS selector accepts or receives requests from client systems and DNS systems through a network interface card 81 connected to the network. Network interface cards are known in the art. The network interface card 81 sends the client requests to a processor 83 through a system bus 85. The processor is configured with firmware or software to resolve the client request.

Detailed Description Text (36):
FIG. 7 also illustrates a block diagram of one embodiment of a combined LB Server

and LBS selector. The combined LBS selector and LB server accepts or receives requests from client systems and DNS systems through a first network interface card 81 and a second network interface card 87 connected to the network. The first network interface card 81 has a different IP address from that of the second network interface card 87. The first network interface card 81 and the second network interface card 87 send the client requests to a processor 83 through a system bus 85 and individually send and receive the client requests from the network. The processor configured with a LBS selector firmware or software and a LB server firmware or software to determine the destination for the client request. Since both the LBS selector software and the LB server software reside on the same computing device, both softwares are able to work together without accessing the network. Therefore, the time needed to determine the destination for the client request is accelerated. Once the specific server is identified, the processor sends an IP address through the system bus back to the second network interface card. The second network interface card sends the server response out into the network and thereby back to the client DNS.

Detailed Description Text (38):
In one embodiment of the load balancing system, the load balancing system, by utilizing DNS, effectively distributes client requests to different servers. For example, the server DNS or the client DNS does not have to automatically send the client request to the LBS selector. In one embodiment, the server DNS and the user DNS are configured to only route client requests for servers that are being "load balanced". Therefore, this allows for only some of the servers to be load balanced and others not to be load balanced, if so desired. For instance, some servers may be access infrequently because they maybe old or not contain the most heavily used applications. As such, there maybe no need for this server and others like it to be "load balanced". Therefore, the load balancing components, the LBS selector and the LB server can be bypassed. This would save the series of operations of sending and resolving client requests by the LBS selectors and LB servers.

Detailed Description Text (39):
Therefore, if the client requests a server not "load balanced", the server DNS can directly provide the IP number of the server to the client system. However, if the client requests a server "load balanced", the server DNS sends the client request to the LBS selector for resolution. Alternatively, if load balancing is desired, e.g. by the client system, the server DNS sends the client request to the LBS selector for resolution and if load balancing is not desired, the server DNS can directly provide the IP number of the server to the client system.

Detailed Description Text (40):
Furthermore, in one embodiment of the load balancing system of the present invention, DNS caching is provided for the resolving of domain names for network components such as the client DNS, the server DNS, the LBS selector, and the LB server. As such, when a domain name is resolved in a client request, any subsequent client request are resolved in the same fashion. For instance, if the client system C1 requests a server S1, the client DNS resolves the server S1 with the IP number 147.244.54.1 and this resolution is cached or temporarily stored within the client DNS. A subsequent client request for server S1 by client system C1 or any other client system resolves to the IP number stored in the cache of the client DNS. Similarly, the server DNS, LB server, and LBS selector or any combination thereof can be configured with DNS caching. DNS caching reduces the amount of overhead, similar to the LB server's use of macro-control, such that the LB server does not have to constantly resolve client requests.

Detailed Description Text (41):
In another embodiment of the load balancing system of the present invention, the network is configured such that the client DNS skips the sending of the client request to the server DNS, but instead to the LB server. In this case, the client DNS and the LB server handle all domain name resolution requests. In other words,

the LB server becomes the primary DNS for the server side of the network. Therefore, the LB server has to handle all the domain name resolution requests to a server even if the server requested not subjected to load balancing. Furthermore, since the LB server acts as the primary DNS, the LBS selector does not have to select a LB server to receive the client request. The client requests are already being sent directly to the LB server.

Detailed Description Text (42):
For instance, the client system transmits a client request, e.g. a domain name resolution request, to the client DNS. If the client DNS cannot resolve the request, the client DNS sends the client request to the LB server. The LB server is on the server side of the network. The LB server receives the forwarded client request from the client DNS. The LB server resolves the client request by examining the client request and the network measurements provided for the servers. Based on the network measurements and the client request, the LB server determines the best server for handling the client request. Once the determination is made by the LB server and a server is selected, the LB server provides an IP number to the client system. Using the provided IP number, the client system directs its request to that IP number.

Detailed Description Text (43):
Also, in another embodiment of the load balancing system of the present invention, the network is configured such that the client system and the servers are in same domain. In other words, there is a single DNS. Since servers and client systems are in the same domain, the server DNS and the client DNS are one in the same. For instance, the client system transmits a client request, e.g. a domain name resolution request, to the client DNS. The client DNS resolves the client request by providing the IP number of the LBS selector. The LBS selector is on the server side of the network. The LBS selector receives the forwarded client request from the client DNS. The LBS selector resolves the client request and provides the IP number of the LB server to the client DNS. The client DNS sends the client request to the LB server in response to the IP number provided by the LBS selector. The LB server examines the client request and the network measurements provided for the servers. Based on the network measurements and the client request, the LB server determines the best server for handling the client request. Once the determination is made by the LB server and a server is selected, the LB server provides an IP number to the client system. Using the provided IP number, the client system directs its request to that IP number. Therefore, a single LB server can be configured to operate within a single DNS.

Detailed Description Text (44):
Referring once more to FIG. 1, although only one LBS selector, LBS selector 15, is depicted, in another embodiment of the present invention, there are more than one LBS selectors. The multiple LBS selectors operate independently from each other and service or receive requests from several client systems. The multiple LBS selectors also communicate with some or all of the LB servers within the network. Since the LBS selectors operate independently from each other, an LBS selector can be assigned to receive request from a subset of client systems. This, in effect, segments the client systems and reduces the amount of network traffic or load on each of the LBS selectors within the network. For instance, LBS selector Z1 is connected with client systems A1-A10 and LBS selector Z2 is connected to client systems B1-B10 instead of only one LBS selector C1 being connected to client systems A1-A10 and client systems B1-B10. Hence, by segmenting or dividing the client systems and associating each segment of client systems with a different LBS selector, the amount of network traffic or load experienced by a single LBS selector is significantly reduced. Similarly, in another embodiment, the server DNS partitions the multiple LBS selectors among the multiple LB servers.

Detailed Description Text (45):
Additionally, with multiple LBS selectors, a second LBS selector can act as a

backup to a first LBS selector. For instance, in a network with two LBS selectors F1 and B1, both LBS selectors can be configured to receive requests from the same client systems for the same LB servers. However, LBS selector F1 is active or operational and LBS selector is inactive although connected to the network. If LBS selector F1 becomes inoperable, LBS selector B1 is activated and thereby quickly replaces the LBS selector F1. As a result, the network is once again operational within a matter of minutes instead of hours or days.

Detailed Description Text (46):
Furthermore, with multiple LBS selectors, the LBS selectors can be configured operate in a sharing mode such that the load or network traffic experienced by any one LBS selector is significantly reduced. For instance, a first LBS selector A1 and a second LBS A2 both receive requests from a specific set of client systems for a specific set of LB servers. However, if the first LBS selector A1 becomes overloaded, e.g. the network traffic on the first LBS selector A1 exceeds a predetermined threshold such as 1M bits/sec, then the first selector A1 stops accepting requests from the client systems. The second LBS selector A2 assumes the handling of the requests from the client systems that was originally being handled by the first LBS selector. Similarly, the LBS selectors can be configured to operate in a sharing mode in a round robin fashion in an effort to reduce the load or network traffic experienced by any one LBS selector.

CLAIMS:

1. A method of load balancing for a network, the network having a plurality of client systems and a plurality of servers, comprising: receiving a client request for a server from a client system; resolving the destination point of the client request; sending the client request to a load balancing server based on the resolved destination point; selecting one server out of a subset of the plurality of servers based on a predetermined criteria; providing a conduit to the client system and the selected server for transmission of the client request; receiving network measurements, the network measurements being a total amount of bytes of data received by each of the subset of the plurality of servers from the plurality of client systems and a total amount of bytes of data sent by each of the subset of the plurality of servers to the plurality of client systems over a predetermined time period and the network measurements received is the predetermined criteria; and dynamically switching between one of two modes based on load experienced by the load balancing server, a first mode being macro-mode and a second mode being micro-mode; and wherein the load balancing server, in micro-mode, monitors each request from the plurality of client systems and, in macro-mode, causes requests from the plurality of client systems to bypass the load balancing server and be sent directly from the plurality of client systems to the selected server.

5. The method of claim 1 further comprising determining the load on each server within the subset of the plurality of servers.

6. The method of claim 1 further comprising sending requests from the client system to the selected server.

8. A load balancing system for a network, the network having a plurality of client systems and a plurality of servers, comprising: at least one load balancing server adapted to be coupled with the network; and at least one load balancing server selector coupled with the at least one load balancing server, the load balancing server selector adapted to receive requests from the client systems and to forward the requests to the at least one load balancing server, the at least one load balancing server selecting one server out of a subset of the plurality of servers based on a predetermined criteria and the requests from the client systems; wherein the at least one load balancing server is further configured to operate in one of two modes, a first mode being macro-mode and a second mode being micro-mode and the at least one load balancing server in micro-mode monitors the requests from the

client systems and a portion of the requests are session specific.

11. The load balancing system of claim 8 wherein the requests from the client systems are domain name resolution requests, requesting to resolve a logical name to an IP address.

15. The load balancing system of claim 8 further comprising a client domain name server, such that the requests from the client systems are sent to the client domain name server to resolve a logical name to an IP address.

16. The load balancing system of claim 15 further comprising a server domain name server, such that the requests from the client domain name server are sent to a server domain name server based upon a predetermined condition to resolve a logical name with an IP address.

17. The load balancing system of claim 16 wherein one of the client systems is a member of a domain; and the predetermined condition is a determination of whether the request from the client system is to a server that is not a member of the domain or not.

18. The load balancing system of claim 16 wherein the predetermined condition is a determination whether the client domain name server is able to resolve the requests from the client systems.

19. A load balancing server for a network, the network having a plurality of client systems and a plurality of servers, comprising: a processor configured to select one server out of a subset of the plurality of servers based on a predetermined criteria and requests from the client systems; a network interface adapted to be coupled with the network and the processor; wherein the network interface receives network measurements, the network measurements being a total amount of bytes of data received by each of the subset of the plurality of servers from the plurality of client systems and a total amount of bytes of data sent by each of the subset of the plurality of servers to the plurality of client systems over a predetermined time period and the network measurements received is the predetermined criteria used by the processor; and wherein the processor is further configured to operate in one of two modes, a first mode being macro-mode and a second mode being micro-mode and the processor in micro-mode monitors the requests from the plurality of client systems and a portion of the requests are session specific.

22. The load balancing server of claim 21 wherein the processor is configured to monitor a total number of requests from the plurality of client systems received by the load balancing server over a predetermined time period, such that the processor switches configuration to send a different IP address to one of the plurality of the client systems, and the different IP address represents a path from the client system to the load balancing server to one of the plurality of servers.

24. The load balancing server of claim 23 wherein the processor is configured to monitor a total number of requests from the plurality of client systems received by the load balancing server over a predetermined time period, and the processor switches configuration to send a different IP address to the client systems representing one of the plurality of servers.

25. A load balancing system for a network, the network having client systems and servers, the system comprising: at least one load balancing server adapted to be coupled with the network; at least one load balancing server selector coupled with the at least one load balancing server, the load balancing server selector adapted to receive requests from the client systems and to forward the requests to the at least one load balancing server, the at least one load balancing server selecting one server out of a subset of the servers based on a predetermined criteria and the requests from the client systems; wherein the at least one load balancing server is

configured to dynamically switch between one of two modes based on load experienced by the at least one load balancing server, a first mode being macro-mode and a second mode being micro-mode; and wherein the at least one load balancing server, in micro-mode, monitors each request from the client systems and, in macro-mode, causes requests from the client systems to bypass the load balancing server and be sent directly from the client systems to the selected server.

27. The system of claim 26 wherein the load experienced comprises computation processing based on a total number of requests from the client systems received by the load balancing server over a predetermined time period to be processed by the load balancing server.

31. The system of claim 30 wherein the predetermined limit is based on a total number of requests from the client systems received by the load balancing server over a predetermined time period to be processed by the load balancing server.

34. The system of claim 33 wherein the predetermined limit is based on a total number of requests from the client systems received by the load balancing server over a predetermined time period to be processed by the load balancing server.

First Hit Fwd Refs

Generate Collection

L9: Entry 5 of 24 File: USPT Dec 16, 2003

DOCUMENT-IDENTIFIER: US 6665702 B1
TITLE: Load balancing

Abstract Text (1):
This invention discloses a method for managing a computer network connected to the Internet through a network connection, such as different Internet Service Providers, including the steps of; sending polling requests through a plurality of routes from a computer network to a remote server computer, receiving replies from the remote server computer corresponding to the polling requests, and measuring proximities of the remote server computer to the computer network based on the received replies.

Application Filing Date (1):
19991220

Brief Summary Text (2):
The present invention relates to computer networks in general, and in particular to load balancing client requests among redundant network servers in different geographical locations.

Brief Summary Text (4):
In computer networks, such as the Internet, preventing a server from becoming overloaded with requests from clients may be accomplished by providing several servers having redundant capabilities and managing the distribution of client requests among the servers through a process known as "load balancing."

Brief Summary Text (5):
In one early implementation of load balancing, a Domain Naming System (DNS) server connected to the Internet is configured to maintain several IP addresses for a single domain name, with each address corresponding to one of several servers having redundant capabilities. The DNS server receives a request for address translation and responds by returning the list of server addresses from which the client chooses one address at random to connect to. Alternatively, the DNS server returns a single address chosen either at random or in a round-robin fashion, or actively monitors each of the servers and returns a single address based on server load and availability.

Brief Summary Text (6):
More recently, a device known as a "load balancer," such as the Web Server Director, commercially available from the Applicant/assignee, has been used to balance server loads as follows. The load balancer is provided as a gateway to several redundant servers typically situated in a single geographical location and referred to as a "server farm" or "server cluster." DNS servers store the IP address of the load balancer rather than the addresses of the servers to which the load balancer is connected. The load balancer's address is referred to as a "virtual IP address" in that it masks the addresses of the servers to which it is connected. Client requests are addressed to the virtual IP address of the load balancer which then sends the request to a server based on server load and availability or using other known techniques.

Brief Summary Text (7):
Just as redundant servers in combination with a load balancer may be used to prevent server overload, redundant server farms may be used to reroute client requests received at a first load balancer/server farm to a second load balancer/server farm where none of the servers in the first server firm are available to tend to the request. One rerouting method currently being used involves sending an HTTP redirect message from the first load balancer/server farm to the client instructing the client to reroute the request to the second load balancer/server farm indicated in the redirect message. This method of load balancing is disadvantageous in that it can only be employed in response to HTTP requests, and not for other types of requests such as FTP requests. Another rerouting method involves configuring the first load balancer to act as a DNS server. Upon receiving a DNS request, the first load balancer simply returns the virtual IP address of the second load balancer. This method of load balancing is disadvantageous in that it can only be employed in response to DNS requests where there is no guarantee that the request will come to the first load balancer since the request does not come directly from the client, and where subsequent requests to intermediate DNS servers may result in a previously cached response being returned with a virtual IP address of a load balancer that is no longer available.

Brief Summary Text (8):
Where redundant server farms are situated in more than one geographical location, the geographical location of a client may be considered when determining the load balancer to which the client's requests should be routed, in addition to employing conventional load balancing techniques. However, routing client requests to the geographically nearest server, load balancer, or server farm might not necessarily provide the client with the best service if, for example, routing the request to a geographically more distant location would otherwise result in reduced latency, fewer hops, or provide more processing capacity at the server.

Brief Summary Text (10):
The present invention seeks to provide novel apparatus and methods for load balancing client requests among redundant network servers and server farms in different geographical locations which overcome the known disadvantages of the prior art as discussed above.

Brief Summary Text (23):
There is also provided in accordance with a preferred embodiment of the present invention a method for determining network proximity, the method including sending from each of at least two servers a UDP request having a starting TTL value to a client at a sufficiently high port number as to elicit an "ICMP port unreachable" reply message to at least one determining one of the servers indicating the UDP request's TTL value on arrival at the client, determining a number of hops from each of the servers to the client by subtract the starting TTL value from the TTL value on arrival for each of the servers, and determining which of the servers has fewer hops of the client, and designating the server having fewer hops as being closer to the client than the other of the servers.

Brief Summary Text (33):
There is further provided in accordance with a preferred embodiment of the present invention a method for managing a computer network connected to the Internet through a plurality of routes or Internet Service Providers, includes the steps of: sending polling requests through a plurality of ISPs from a computer network to a remote server computer, receiving replies from the remote server computer corresponding to the polling requests, and measuring proximities of the remote server computer to the computer network based on the received replies.

Brief Summary Text (41):
There is also provided in accordance with yet another preferred embodiment of the present a method for managing a computer network connected to the Internet through

a plurality of ISPs, includes the steps of: receiving a request from a client within a computer network directed to a remote server computer, looking up a table entry within a proximity table indexed by an address related to the remote server computer, the tables entries of the proximity table containing ratings for a plurality of ISPs, and selecting one of the plurality of ISPs through which to route the client request, based on the ratings within the table entry looked up in the proximity table.

Brief Summary Text (43):
Still further in accordance with a preferred embodiment of the present invention, the table entries contain the best three choices for ISPs through which to route the client request, and wherein the selecting step selects the best ISP, from among the best three choices for ISPs, that is available and not overloaded.

Brief Summary Text (45):
Further in accordance with a preferred embodiment of the present invention the plurality of ISPs assign respective IP addresses to the computer network, and wherein the method further comprises the step of setting the source IP address of the client request corresponding to the selected ISP.

Brief Summary Text (46):
Moreover in accordance with a preferred embodiment of the present invention the method also includes the step of routing the client request through the selected ISP. Preferably the plurality of ISPs assign respective IP addresses to the computer network, and the routing step designates a source IP address for the client request corresponding to the selected ISP.

Brief Summary Text (47):
The computer network may further be a private network, visible externally through a network address translation. Preferably the method may also include the steps of receiving a response from the remote server directed to the source IP address designated for the client request, and translating the source IP address designated for the client address to the IP address for the client within the private network.

Brief Summary Text (48):
There is further provided in accordance with yet another preferred embodiment of the present invention a network management system for managing a computer network connected to the Internet through a plurality of ISPs, including a network controller sending polling requests through a plurality of ISPs from a computer network to a remote server computer, and receiving replies from the remote server computer corresponding to the polling requests, and a proximity analyzer measuring proximities of the remote server computer to the computer network based on the replies.

Brief Summary Text (54):
There is also provided in accordance with another preferred embodiment of the present invention, a network management system for managing a computer network connected to the Internet through a plurality of ISPs, including a network controller receiving a client request from within a computer network directed to a remote server computer, and selecting one of a plurality of ISPs through which to route the client request, and a data manager looking up a table entry within a proximity table indexed by an address related to the remote server computer, the tables entries of the proximity table containing ratings for a plurality of ISPs. The network controller may also select one of the plurality of ISP based on the rating within the table entry looked up in the proximity table.

Brief Summary Text (56):
Still further in accordance with a preferred embodiment of the present invention, the table entries contain the best three choices for ISPs through which to route

the client request, and the network controller selects the best ISP, from among the best three choices for ISPs, that is available and not overloaded. Preferably the network controller also determines whether or not an ISP is overloaded based upon a user-configurable load threshold.

Brief Summary Text (57):
Additionally in accordance with a preferred embodiment of the present invention, the network controller selects an ISP based on current load, in the event that all three of the best three choices for ISP are unavailable or overloaded The plurality of ISPs may assign respective IP addresses to the computer network, the network controller designates a source IP address for the client request corresponding to the selected ISP.

Brief Summary Text (58):
Moreover in accordance with a preferred embodiment of the present invention, the network controller routes the client request through the selected ISP. Preferably the computer network is a private network, visible externally through a network address translation, and the network controller receives a response from the remote server directed to the source IP address designated for the client request, the system further comprising a network address translator translating the source IP address designated for the client address to the IP address for the client within the private network.

Brief Summary Text (59):
There is also provided in accordance with yet another preferred embodiment of the present invention a method for managing a computer network connected to the Internet through a plurality of ISPs, including the steps of receiving a DNS resolution query from a remote computer for a domain name within a computer network, sending polling requests through a plurality of ISPs from the computer network to the remote computer, receiving replies from the remote computer corresponding to the polling requests, and measuring proximities of the remote computer to the computer network based on the replies.

Brief Summary Text (72):
Further in accordance with a preferred embodiment of the present invention the plurality of ISPs assign respective IP addresses to the computer network and wherein said network controller sets the source IP address of the client request corresponding to the selected ISP.

Brief Summary Text (76):
There is also provided in accordance with yet another preferred embodiment of the present invention a network management system for managing a computer network connected to the Internet through a plurality of ISPs, including a network controller receiving a DNS resolution query from a remote computer for a domain name within a computer network, sending polling requests through a plurality of ISPs from the computer network to the remote computer, and receiving replies from the remote computer corresponding to the polling requests, and a proximity analyzer measuring proximities of the remote computer to the computer network via the plurality of ISPs, based on the replies.

Detailed Description Text (3):
Typical operation of the triangulation load balancing system of FIGS. 1A-1C is now described by way of example. As is shown more particularly with reference to FIG. 1A, a client 26, such as any known computer terminal configured for communication via network 14, is shown sending a request 28, such as an FTP or HTTP request, to LB1 whose virtual IP address is 100.100.1.0. In accordance with network transmission protocols, request 28 indicates the source IP address of the requestor, being the IP address 197.1.33.5 of client 26, and the destination IP address, being the virtual IP address 100.100.1.0 of LB1. LB2 preferably periodically sends a status report 30 to LB1, the virtual IP address 100.100.1.0 of

LB1 being known in advance to LB2. Status report 30 typically indicates the availability of server farm 12 and provides load statistics, which LB1 maintains.

Detailed Description Text (4):
LB2 is preferably capable of having multiple virtual IP addresses as is well known. It is a particular feature of the present invention for LB2 to designate a currently unused virtual IP address, such as 200.100.1.1, for LB1's use and store the mapping between the IP address of LB1 and the designated IP address in a triangulation mapping table 32, as is shown more particularly with reference to FIG. 1B. The designated address is referred to herein as the triangulation address and may be preconfigured with LB1 or periodically provided to LB1 from LB2. LB1 preferably maintains in a client mapping table 36 a mapping of the IP address 197.1.33.5 of client 26 and the triangulation address 200.100.1.1 of LB2 to which client 26's requests may be redirected.

Detailed Description Text (5):
As shown in the example of FIG. 1A, server status table 22 of LB1 indicates that no servers in server farm 10 are available to service client 26's request, but indicates that server farm 12 is available. Having decided that client 26's request should be forwarded to LB2, in FIG. 1C LB1 substitutes the destination IP address of request 28 with the virtual IP address 200.100.1.1 of LB2 which is now mapped to the IP address of client 26 as per client mapping table 36 and sends an address-modified client request 38 to LB2. LB2, upon receiving request 38 at its virtual IP address 200.100.1.1, checks triangulation mapping table 32 and finds that virtual IP address 200.100.1.1 has been designated for LB1's use. LB2 therefore uses the virtual IP address 100.100.1.0 of LB1 as per triangulation mapping table 32 as the source IP address of an outgoing response 40 that LB2 sends to client 26 after the request has been serviced by one of the servers in server arm 12 selected by LB2. It is appreciated that response 40 must appear to client 26 to come from LB1, otherwise client 26 will simply ignore response 40 as an unsolicited packet. Client 26 may continue to send requests to LB1 which LB1 then forwards requests to LB2 at the designated triangulation address. LB2 directs requests to an available server and sends responses to client 26 indicating LB1 as the source IP address.

Detailed Description Text (7):
Typical operation of the network proximity load balancing system of FIGS. 2A-2F is now described by way of example. As is shown more particularly with reference to FIG. 2A, client 26 is shown sending request 28, such as an FTP or HTTP request to LB1 whose virtual IP address is 100.100.1.0. LB1 preferably maintains a proximity table 54 indicating subnets and the best server farm site or sites to which requests from a particular subnet should be routed. Determining the "best" site is described in greater detail hereinbelow.

Detailed Description Text (8):
Upon receiving a request, LB1 may decide to service the request or not based on normal load balancing considerations. In any case, LB1 may check proximity table 54 for an entry indicating the subnet corresponding to the subnet of the source IP address of the incoming request. As is shown more particularly with reference to FIG. 2B, if no corresponding entry is found in proximity table 54, LB1 may send a proximity request 56 to LB2, and LB3, whose virtual IP addresses are known in advance to LB1. Proximity request 56 indicates the IP address of client 26.

Detailed Description Text (9):
A "network proximity" may be determined for a requester such as client 26 with respect to each load balancer/server farm by measuring and collectively considering various attributes of the relationship such as latency, hops between client 26 and each server farm, and the processing capacity and quality of each server farm site. To determine comparative network proximity, LB1, LB2, and LB3 preferably each send a polling request 58 to client 26 using known polling mechanisms. While known polling mechanisms included pinging client 26, sending a TCP ACK message to client

26 may be used where pinging would otherwise fail due to an intervening firewall or NAT device filtering out a polling message. A TCP ACK may be sent to the client's source IP address and port. If the client's request was via a UDP connection, a TCP ACK to the client's source IP address and port 80 may be used. One or both TCP ACK messages should bypass any intervening NAT or firewall and cause client 26 to send a TCP RST message, which may be used to determine both latency and TTL. While TTL does not necessarily indicate the number of hops from the client to the load balancer, comparing TTL values from LB1, LB2, and LB3 should indicate whether it took relatively more or less hops.

Detailed Description Text (10):
Another polling method involves sending a UDP request to a relatively high port number at the client, such as 2090. This request would typically be answered: with an "ICMP port. unreachable" reply which would indicate the TTL value of the UDP request on arrival at the client. Since the starting TTL value of each outgoing UDP request is known, the actual number of hops to the client may be determined by subtracting the TTL value on arrival at the client from the starting TTL value. A combination of pinging, TCP ACK, UDP, TCP SYN, and other polling techniques may be used since any one polling request might fail.

Detailed Description Text (11):
Client 26 is shown in FIG. 2D sending a polling response 60 to the various polling requests. The responses may be used to determine the latency of the transmission, as well as the TTL value. LB2 and LB3 then send polling results 62 to LB1, as shown in FIG. 2E. The polling results may then be compared, and LB1, LB2, and LB3 ranked, such as by weighting each attribute and determining a total weighted value for each server farm. Polling results may be considered together with server farm capacity and availability, such as may be requested and provided using known load balancing reporting techniques or as described hereinabove with reference to FIGS. 1A and 1B, to determine the server farm site that is "closest" to client 26 and, by extension, the client's subnet, which, in the example shown, is determined to be LB2. For example, the closest site may be that which has the lowest total weighted value for all polling, load, and capacity results. LB1 may then store the closest site to the client/subnet in proximity table 54.

Detailed Description Text (12):
As was described above, a load balancer that receives a request from a client may check proximity table 54 for an entry indicating the subnet corresponding to the subnet of the source IP address of the incoming request. Thus, if a corresponding entry is found in proximity table 54, the request is simply routed to the location having the best network proximity. Although the location having the best network proximity to a particular subnet may have already been determined, the load balancer may nevertheless decide to forward an incoming request to a location that does not have the best network proximity should a load report received from the best location indicate that the location is too busy to receive requests. In addition, the best network proximity to a particular subnet may be periodically redetermined, such as at fixed times or after a predetermined amount of time has elapsed from the time the last determination was made.

Detailed Description Text (13):
As is shown more particularly with reference to FIG. 2F, once the closest site for client 26 has been determined, client 26 may be redirected to the closest site using various methods. If a DNS request is received from client 26, LB1 may respond with LB2's address. If an HTTP request is received from client 26, HTTP redirection may be used. Alternatively, regardless of the type of request received from client 26, triangulation as described hereinabove with reference to FIGS. 1A-1C may be used.

Detailed Description Text (19):
In turn, as illustrated in FIG. 3E, content router 145 sends requests issued from

client 160 via router 135, and indicates a source IP address of 30.1.1.1 with each such request, which is the IP address associated with router 135 from within the range of IP addresses allocated by ISP 120.

Detailed Description Text (21):
Reference is now made to FIG. 4A, which illustrates a preferred embodiment of the present invention used to resolve incoming DNS requests for a multi-homed network architecture. Server 170 is assigned IP address 10.3.3.3 within a private multi-homed network, similar to the network illustrated in FIG. 3A. Each of ISPs 115, 120 and 125 assigns a range of IP addresses to the multi-homed network A DNS request for resolution of a domain name is issued from a client 175 with IP address 192.115.90.3. The DNS request has a source IP address of 192.115.90.3 and a destination IP address of 20.1.1.1. As such, it arrives at content router 145 via router 130.

Detailed Description Text (22):
FIG. 4B indicates a NAT mapping table 180, showing that the private IP address 10.3.3.3 for server 170 is translated to IP addresses 20.3.3.3, 30.3.3.3 and 40.3.3.3, respectively, by routers 130, 135 and 140. Content router 145 looks up the subnet entry 192.115.90 in proximity table 155, and identifies router 135 as the first choice for best proximity between server 170 and client 175. In resolving the DNS request, content router 145 accordingly provides 30.3.3.3 as the IP address for the domain name server, even though the original request indicated a destination IP address of 20.1.1.1. This ensures that requests from client 175 are sent to server 170 with a destination IP address of 30.3.3.3, which in turn ensures that the client requests are transmitted through ISP 120.

Detailed Description Text (24):
Referring back to FIG. 3F, suppose for example that ISP 120 is unavailable, and that content router 145 routes the outgoing client request through ISP 125 instead of through ISP 120. In accordance with a preferred embodiment of the present invention, content router 145 routes the outgoing request through ISP 125 and labels the outgoing request with a source IP address of 40.1.1.1. Had content router 145 used ISP 125 but indicated a source IP address of 30.1.1.1, the response from server 150 would be directed back through ISP 125, and not be able to get through to client 160.

Detailed Description Text (25):
Similarly, referring back to FIG. 4B, suppose for example that ISP 120 is unavailable, and that content router 145 resolves the DNS request with IP address 40.3.3.3 instead of IP address 30.3.3.3. This ensures that client 175 directs it requests through ISP 125, and avoids any blockage at ISP 120.

CLAIMS:

1. A method for managing a computer network connected to the Internet through a plurality of routes, comprising the steps of: receiving a request from a client within a client computer network directed to a remote server computer within a second computer network; looking up a table entry within a proximity table indexed by an address related to the remote server computer, the tables entries of the proximity table containing ratings for a plurality of routes between the client computer network and the second computer network; and selecting one of the plurality of routes through which to route the client request, based on the ratings within the table entry looked up in the proximity tables, wherein the plurality of routes assign respective IP addresses to the computer network, and wherein the method further comprises the step of setting the source IP address of the client request corresponding to the selected route on the client side.

3. The method of claim 1 wherein the table entries contain the best three choices for routes through which to route the client request, and wherein said selecting

step selects the best route, from among the best three choices for routes, that is available and not overloaded.

7. The method of claim 6 further comprising the steps of: receiving a response from the remote server directed to the source IP address designated for the client request; and translating the source IP address designated for the client request to the IP address for the client within the private network.

8. A network management system for managing a computer network connected to the Internet through a plurality of routes, comprising: a network controller receiving a client request from within a client computer network directed to a remote server computers, within a second computer network and selecting one of a plurality of routes through which to route the client request; and a data manager looking up a table entry within a proximity table indexed by an address related to the remote server computer, the tables entries of the proximity table containing ratings for a plurality of routes, between the client computer network and the second computer network and wherein said network controller selects one of the plurality of routes based on the ratings within the table entry looked up in the proximity tables, wherein the plurality of routes assign respective IP addresses to the computer network, and wherein said network controller sets the source IP address of the client request corresponding to the selected route on the client side.

10. The network management system of claim 8 wherein the table entries contain the best three choices for routes through which to route the client request, and wherein said network controller selects the best route, from among the best three choices for routes, that is available and not overloaded.

14. The network management system of claim 13 wherein said network controller receives a response from the remote server directed to the source IP address designated for the client request, the system further comprising a network address translator translating the source IP address designated for the client request to the IP address for the client within the private network.

L9: Entry 6 of 24 File: USPT Dec 2, 2003

DOCUMENT-IDENTIFIER: US 6658479 B1
TITLE: Load-balanced anycasting and routing in a network

Application Filing Date (1):
20000630

Brief Summary Text (9):
With server load-balancing, a computation is assigned to a server based on the current computational load on various other servers, possibly with provisions for failover to handle the loss of a server or communication link. Commercial products typically use round robin, least number of connections, weighted priority, and server load or response time to determine how to distribute traffic among servers, but treat server load and network load separately.

Detailed Description Text (4):
Load-balanced anycasting and routing is premised on allowing user requests to refer to services in a network, rather than the servers where the services are provided, and that a given service is available at multiple servers. A service can be distributed across multiple servers, with subsets of the nodes in the network providing a particular processing step for the service. Each service may comprise one or more routes, and the sequence of processing steps for every route for that service is fixed and known in advance.

First Hit Fwd Refs

Generate Collection

L9: Entry 7 of 24 File: USPT Oct 29, 2002

DOCUMENT-IDENTIFIER: US 6473791 B1
TITLE: Object load balancing

Application Filing Date (1):
19980817

CLAIMS:

5. The method of claim 1 wherein a selected replaceable load balancing engine designated for an object class determines when monitored performance of the host server computer is preferred over monitored performance of other server computers in the server group.

20. In a router computer, a load balancing service for distributing an object creation request comprising an object creation request characteristic among a target group comprising a plurality of server computers, the load balancing service comprising: a host server computer selection means for selecting a host server computer to host the object creation request, the host server computer selection means operative to accept a supplied object creation request characteristic and specify a selected host server computer, the specified host server computer selected by the host server computer selection means from the target group based on the specified object creation request characteristic; a system service for routing the component creation request to a server computer in the target group, the system service operative to accept the component creation request comprising the object creation request characteristic, supply the object creation request characteristic to the host server computer selection means, accept from the host server computer selection means a specified selected host server computer and route the object creation request thereto.

21. The load balancing service of claim 20 wherein the object creation request characteristic of the object creation request is a class identifier, and the host server computer selection means specifies a server computer having an activated instantiated object of the class identifier.

22. The load balancing service of claim 20 wherein the object creation request characteristic of the object creation request is a client identity identifying a client computer, and the host server computer selection means specifies a server computer to which object creation requests for the client computer have previously been routed based on the previously-routed requests.

23. The load balancing service of claim 20 wherein the object creation request characteristic is selected from the group consisting of a class identifier of the object creation request, a client computer identity of a computer issuing the object creation request, and a process identity of a process issuing the object creation request.

29. In a computer network, an object creation architecture for balancing a load of object creation requests among a plurality of server computers, the architecture comprising: a routing table comprising a plurality of stored object class identifiers, wherein at least one stored object class identifier is associated with

a server computer; at a router computer, a load balancing service responsive to a supplied object class identifier in an object creation request from a client program on a client computer and operative to select a server associated with the supplied object class identifier in the routing table, the load balancing service further operative to route the object creation request to an object creation service at the selected server computer; at the selected server computer, an object creation service responsive to the object creation request from the load balancing service and operative to create a server object of an object class associated with the supplied identifier and further operative to assemble a stub with the server object, the stub operative to monitor calls to the server object to observe and store in a metric data store at the selected server computer a performance value, the performance value indicative of performance at the selected server computer according to a processing metric; at the router computer, a metric collector operative to retrieve the observed performance value from the metric data store and integrate the performance value into a collective metric data store, wherein the collective metric data store comprises metric data from plural server computers; and a load balancing engine at the router computer operative to consult the collective metric data store and associate in the routing table an object class identifier with a server having a performance value determined superior according to the processing metric by the load balancing engine.

31. In a computer network comprising a router computer, a plurality of server computers and a plurality of client computers, an architecture for balancing a load of computer object processing among the server computers, the architecture comprising: a routing table at the router computer associating object classes with server computers; a monitor at a server computer, the monitor operative to intercept a reference to an instantiated first software object of a monitored object class to transparently conduct and record a processing metric observation, the monitor further operative to send a processing metric value based on the processing metric observation and indicative of performance at the server computer; a load balancing service at the router computer, the load balancing service operative to receive a client computer request to create a second object of the monitored object class and route the request to a selected server associated with the monitored object class in the routing table, the load balancing service responsive to the processing metric value sent by the monitor to associate a server having a favorable processing metric value with the monitored object class in the routing table; and an object creation service at the selected server operative to receive the request from the load balancing service and create an object of the monitored object class.

34. In a computer network having a router computer and a plurality of server computers in a target group, a method for balancing object processing among the plurality of server computers, the method comprising: conducting plural processing performance metric observations associated with a software object class at a server computer; periodically blending the observations into a representative value indicative of performance at the server computer; periodically transferring the representative value from the server computer to a router computer to provide plural successive representative values to the router computer, wherein transferring is sufficiently delayed to facilitate blending a number of observations to dampen variance in the successive representative values; receiving at a router computer the plural representative values from the server computer and plural representative values from at least one other server computer in the target group; and routing resource requests received by the router computer to a server computer in the target group having a representative value indicative of more favorable performance than another server computer in the target group.

38. A load balancing service for balancing object processing among a plurality of server computers by accommodating object creation requests from a plurality of client programs executing on a plurality of client computers, the load balancing service comprising: at a client computer, a configuration database for associating

object classes with remote computers, at least one object class in the configuration database associated with a router computer; at the client computer, an operating system service operative to receive an object creation request comprising an object class, the operating system service further operative to direct the object creation request to a computer associated with the object class in the configuration database; at the router computer, a routing table for associating an object class with a server computer; at the router computer, a routing service operative to receive the object creation request from the client computer and route the request to a selected server computer associated with the request's object class in the routing table; at the selected server computer, a class instance creator operative to receive the object creation request and create an object of the request's object class; an operating system service for providing a wrapper around the object, the wrapper comprising a method invocation service, the method invocation service operative to receive invocations of a method of the object from a client program and forward the invocation to the object, the method invocation service further operative to observe execution of the method by the object to produce an object class method performance observation, the object class method performance observation associated with the object class of the object and indicative of the method's performance according to a processing metric; an observation collection service at the router computer operative to collect and store the object class method performance observation and at least one other object class method performance observation in a collective observation store from a target group, the target group comprising the selected server computer and other plural server computers; and a load balancing engine at the router computer operative to evaluate the object class method performance observations from the target group to associate a favorable server computer in the target group with a selected object class in the routing table, the evaluated observations associated with the selected object class in the collective observation store, the favorable server computer having a more favorable object class method performance observation than another server in the target group according to the processing metric.

40. A computer-readable medium having stored thereon a data structure for routing object creation requests from a remote client computer, the data structure comprising: identifiers indicative of an object class; and a server identifier associated in the data structure with a selected one of the identifiers indicative of an object class, the server identifier indicative of a server computer providing to a router computer favorable processor metric observations for processing objects of the object class indicated by the selected one of the identifiers indicative of an object class; wherein the data structure is an accelerated routing table comprising a lookup table associating a server identifier with a hash function of an identifier indicative of an object class.

DOCUMENT-IDENTIFIER: US 6327622 B1
TITLE: Load balancing in a network environment

Application Filing Date (1):
19980903

Brief Summary Text (3):
In many computing environments, clients (e.g., computer systems and users) connect to servers offering a desired application or service--such as electronic mail or Internet browsing. One computer server may, however, only be capable of efficiently satisfying the needs of a limited number of clients. In such a case, an organization may employ multiple servers offering the same application or service, in which case the client may be connected to any of the multiple servers in order to satisfy the client's request.

Brief Summary Text (4):
A service offered simultaneously on multiple servers is often termed "replicated" in recognition of the fact that each instance of the service operates in substantially the same manner and provides substantially the same functionality as the others. The multiple servers may, however, be situated in various locations and serve different clients. Application programs may also operate simultaneously on multiple servers, with each instance of an application operating independently of, or in concert with, the others. In order to make effective use of an application or replicated service offered by multiple servers (e.g., to satisfy clients' requests), there must be a method of distributing clients' requests among the servers and/or among the instances of the application or service. This process is often known as load balancing. Methods of load balancing among instances of a replicated service have been developed, but are unsatisfactory for various reasons.

Brief Summary Text (5):
In one method of load balancing a replicated service, clients' requests are assigned to the servers offering the service on a round-robin basis. In other words, client requests are routed to the servers in a rotational order. Each instance of the replicated service may thus receive substantially the same number of requests as the other instances. Unfortunately, this scheme can be very inefficient.

Brief Summary Text (6):
Because the servers that offer the replicated service may be geographically distributed, a client's request may be routed to a relatively distant server, thus increasing the transmission time and cost incurred in submitting the request and receiving a response. In addition, the processing power of the servers may vary widely. One server may, for example, be capable of handling a larger number of requests or be able to process requests faster than another server. As a result, a more powerful server may periodically be idle while a slower server is over-burdened.

Brief Summary Text (7):
In another method of load balancing, specialized hardware is employed to store

information concerning the servers hosting instances of a replicated service. In particular, according to this method information is stored on a computer system other than the system that initially receives clients' requests. The stored information helps identify the server having the smallest load (e.g., fewest client requests). Based on that information, a user's request is routed to the least-loaded server. In a web-browsing environment, for example, when a user's service access request (e.g., a connection request to a particular Uniform Resource Locator (URL) or virtual server name) is received by a server offering Domain Name Services (DNS), the DNS server queries or passes the request to the specialized hardware. Based on the stored information, the user's request is then forwarded to the least-loaded server offering the requested service.

Brief Summary Text (8):
This method is also inefficient because it delays and adds a level of complexity to satisfying access requests. In particular, one purpose of a DNS server is to quickly resolve a client's request for a particular service to a specific server (e.g., a specific network address) offering an instance of the service. Requiring the DNS server to query or access another server in order to resolve the request is inefficient and delays the satisfaction of the request.

Brief Summary Text (9):
In yet other methods of balancing requests among multiple instances of a replicated service, client requests are randomly assigned to a server or are assigned to the closest server. Random assignment of client requests suffers the same disadvantages as a round-robin scheme, often causing requests to be routed to geographically distant servers and/or servers that are more burdened than others. This naturally results in unnecessary delay. Simply assigning requests to the closest server may also be inefficient because a faster response may be available from a server that, although further from the client, has less of a load.

Brief Summary Text (12):
In one embodiment of the invention a system and methods are provided for balancing client (e.g., user) requests among multiple instances of an application (e.g., application program or replicated service) in accordance with a selected policy. In this embodiment, each instance of the load-balanced application executes on a separate computer server.

Brief Summary Text (13):
A load balancing policy is selected for distributing the client requests among the multiple servers and instances of the application and, at periodic intervals, a "preferred" server is identified in accordance with the policy. Illustratively, the selected policy reflects or specifies one or more application-specific factors or characteristics to be considered in choosing the preferred server. Client requests are routed to the preferred server until such time as a different server is preferred. A selected load balancing policy may be replaced while the application continues operating.

Brief Summary Text (14):
Other exemplary policies reflect preferences for the least-loaded instance of the application or the instance having the fastest response time. The least-loaded instance may be that which has the fewest connected clients and/or the fewest pending client requests. In another policy, where the closest instance of the application is favored, the preferred server may be the server that can be reached in the fewest network hops or connections. Another illustrative policy favors the server and/or the instance with the greatest throughput (e.g., the highest number of client requests satisfied in a given time period).

Brief Summary Text (15):
Depending upon the selected policy, status objects (e.g., agents, modules or other series of executable instructions) are configured to collect these various pieces

of information from each instance of the application that is being load-balanced (and/or its server). Status objects in one embodiment of the invention thus retrieve application-specific information (e.g., number and/or type of pending client requests) and/or information concerning a server's general status (e.g., its distance from another network entity). Illustratively, each instance of a load-balanced application is associated with its own status object(s). In one embodiment of the invention multiple status objects having different functions are associated with one instance.

Brief Summary Text (18):
In an embodiment of the invention in which clients access the application through a central server such as a Domain Name Services (DNS) server, a specialized updater object updates a lookup table (e.g., a DNS zone file) to identify the preferred server (e.g., by its network address or an alias). In this embodiment the lookup table is used to resolve a virtual server name (e.g., a virtual identity of the application) to a particular server offering an instance of the application. When a client requests an application via a virtual name, the central server directs the request to the server indicated in the lookup table (i.e., the preferred server). The specialized object is thus configured to update the lookup table (or other data structure) or otherwise cause the direction or re-direction of load-balanced requests to the preferred server.

Drawing Description Text (2):
FIG. 1 is a block diagram depicting an illustrative environment in which an embodiment of the present invention may be implemented to load balance client requests among multiple instances of an application.

Drawing Description Text (3):
FIG. 2 is a block diagram depicting a method of balancing client requests among application instances in accordance with an embodiment of the present invention.

Drawing Description Text (4):
FIG. 3 is a block diagram depicting a method of balancing client requests among geographically dispersed application instances in accordance with an embodiment of the present invention.

Detailed Description Text (3):
In particular, illustrative embodiments of the invention are described in the context of applications such as a database management system (DBMS), electronic mail, or web browsing. Various embodiments of the invention may therefore involve the use of a central server, such as a Domain Name Services (DNS) server, to resolve an access request for an application into an address of a physical machine such as a computer server. One skilled in the art will recognize that the present invention is not limited to the applications described herein or the use of a DNS server, and may be readily adapted to other applications and services for which load balancing is appropriate.

Detailed Description Text (6):
In a present embodiment of the invention, information concerning instances of an application (e.g., an application program or replicated service) operating on multiple computer servers is collected and analyzed to identify a "preferred" server. Illustratively, a preferred server is the server to which client requests for the application are to be routed for processing. A preferred server is identified on a regular or periodic basis, and may be the same as or different from the server previously identified. By periodically changing the preferred server, client requests are load-balanced between the participating servers. Individual clients may thus be routed to, and their requests (e.g., database access, send electronic mail, browse a web page) satisfied by, any of the multiple servers.

Detailed Description Text (7):

The information that may be collected concerning an instance of the program illustratively includes its response time for a client request, its operational status (e.g., whether it is up or down), the number of clients connected to the instance, the number of client requests pending with the instance, its throughput (e.g., number of client requests handled in a period of time), etc. Information concerning the status or performance of the host servers themselves (e.g., load, capacity, distance from a central server) may also be collected and analyzed as part of the process of choosing a preferred server.

Detailed Description Text (8):
Illustratively, a central server that distributes client requests for the application among the various instances uses a lookup table or other data structure or means to store an identifier of the current preferred server. The central server is, in one embodiment of the invention, a Domain Name Services (DNS) server. In this embodiment, the application is exposed (e.g., identified) as a virtual server name to which clients connect and which the DNS resolves to an address of one of the multiple servers operating an instance of the application.

Detailed Description Text (9):
The specific information that is collected (from the various application instances and, possibly, the host servers) is determined by a load balancing policy that may be selected by a system manager or administrator. The preferred server is then selected by analyzing the collected information. Thus, in one illustrative policy, the preferred server is the server offering the application instance that is least-loaded (e.g., has the fewest pending client requests or fewest connected clients). In another illustrative policy, the preferred server is the server closest to the central server.

Detailed Description Text (12):
Generating application-specific status objects or modules illustratively allows the collection of any information that could form the basis for load balancing client requests. For example, to load-balance a database application, it may be desirable to determine the number of users being serviced by each instance of the application, the number of users that have accessed an instance, or the number of access requests that are pending with or that have been processed by each instance. The information gathered by the application-specific status objects is used by other objects and/or modules in the load-balancing framework in order to determine a preferred server.

Detailed Description Text (13):
FIG. 1 is a block diagram depicting an illustrative environment in which an embodiment of the invention may be implemented to balance client requests among multiple instances of an application executing on multiple servers. In the embodiment central server 100 is a computer system that receives information from the various application instances (and possibly the servers hosting the application instances) and routes requests from clients such as client 120 to a preferred server. In one embodiment of the invention, central server 100 is a DNS server. Back-end or host servers 110, 112 and 114 each offer one or more instances of application 104, represented by the numerals 104a, 104b and 104c. Servers 110, 112 and 114 may be geographically or logically separated from one another.

Detailed Description Text (16):
In the environment of FIG. 1, when client 120 attempts to connect to application 104, the access request is received by central server 100. Central server 100, through lookup table 102, identifies a preferred server offering an instance of program 104 and routes the client request accordingly. The server identified in lookup table 102 may be determined according to a load-balancing policy, as discussed below. Further, the server identified in lookup table 102 is updated or changed from time to time in accordance with the selected policy in order to distribute client requests among the instances of the application.

Detailed Description Text (17):
In a present embodiment of the invention, information reflecting the status or operation of application instances 104a, 104b and 104c (and/or servers 110, 112 and 114) is collected and analyzed on a regular or periodic basis. The information that is collected is identified in a load balancing policy that identifies one or more factors or pieces of information to be used to identify a "preferred" server to which client requests for application 104 are to be routed. Different policies thus require different information to be collected from the application instances, and the active policy can be changed during load balancing.

Detailed Description Text (25):
The configuration of the status objects (e.g., the data they collect) depends upon the policy that has been selected for choosing a preferred server. For example, where the selected policy requires choosing the least-loaded server (e.g., the server having the least-loaded instance of the application), a status object may be configured to retrieve the number of pending client requests or number of connected clients. As another example, status objects 200, 202 and 204 may be configured to retrieve a response time or throughput of their associated application instances.

Detailed Description Text (27):
Illustratively, status objects 200, 202 and 204 communicate with or access application instances 104a, 104b and 104c in accordance with an application-specific API. Each status object also illustratively performs a single function or retrieves a single piece of application-specific information. In alternative embodiments of the invention, however, a single status object may perform multiple functions or produce multiple pieces of information. For example, in one alternative embodiment, a status object may retrieve multiple pieces of information concerning an application instance's load (e.g., number of connected clients, number of pending requests). The multiple pieces of information may then be combined (e.g., via a specified formula or function) to produce a single value or representation of the instance's load.

Detailed Description Text (31):
Although each IMO is associated with only one status object and one application instance in the illustrated embodiment, in an alternative embodiment of the invention an IMO may collect data from multiple status objects. In this alternative embodiment, for example, an IMO may interface with one status object to determine the response time of an application instance or server and another status object to determine the load on the instance or server.

Detailed Description Text (35):
The data collected by RMO 220 from the various IMOs is analyzed in accordance with the selected policy and a preferred server is identified. Illustratively, updater object 230 performs the analysis and selection of a preferred server. As discussed above, the preferred server may, for example, be the one having the application instance with the fastest response time, the fewest pending client requests, the greatest capacity for client requests, etc. Illustratively, RMO 220 maintains a data structure (e.g., array, vector, table, database) identifying each application instance and/or server that is being load-balanced, along with one or more values or other indicators or summaries of the collected information concerning each application instance.

Detailed Description Text (37):
In the embodiment of the invention depicted in FIG. 2, RMO 220 retrieves the collected data and updater object 230 updates the lookup table on a periodic basis. The identity of the preferred server may thus change over time so that the client requests are distributed among all active application instances.

Detailed Description Text (38):

The status objects, IMOs, RMO and updater object may be considered to comprise a load-balancing framework for distributing client requests among various instances of an application. As one skilled in the art will recognize, the different objects within the framework may be distributed among the servers hosting application instances, a central server, and other entities such as intermediate servers.

Detailed Description Text (43):
In another alternative embodiment of the invention, load balancing among instances of an application is performed among multiple participating servers wherein one or more of the servers are segregated (e.g., situated in a remote location and/or within a server farm). Within the group of segregated servers, a "local" load balancing policy may be implemented for distributing all client requests sent to the group and/or to a specific member of the group. In this alternative embodiment, the segregated servers may be considered a single entity for the purposes of a "global" load balancing policy specifying the manner in which client requests for the application are to be distributed among participating servers. The global and local policies need not be equivalent (e.g., the global policy may require selection of the closest server (or group of servers) while the local policy may require the least-loaded server or application instance).

Detailed Description Text (44):
With reference now to FIGS. 4 and 5, an illustrative method of load balancing between multiple instances of an application is depicted. In the illustrated method, a central server (e.g., a DNS Server) resolves client requests for a virtual name by which the application is known into an identifier of a preferred server offering an instance of the application. Each instance of the application illustratively operates on a separate server and is modified to produce application-specific information needed to choose the preferred server.

Detailed Description Text (47):
Illustrative policies in a present embodiment of the invention focus upon the status or availability of the various instances of the application. Such policies reflect preferences for the least loaded instance, the instance with the fastest response time or throughput, the instance with the fewest connected clients, etc. For example, where access requests for a database management system (DBMS) are load balanced, illustrative policies may include routing requests to the server on which the fewest DBMS requests have been processed or the server having the fewest connected users or the fewest unfulfilled processing or access requests. For each application for which requests are load-balanced, separate policies may be employed.

Detailed Description Text (49):
In general, the selected policy reflects whichever aspect or aspects of the load-balanced application form the basis for distributing client requests among the various instances of the application and/or the servers hosting the application instances. The information reflecting these aspects is periodically captured for each instance by status objects working in close cooperation with the application instances.

Detailed Description Text (57):
With reference now to FIG. 5, illustrative registration and monitoring stages of the illustrated method are depicted. For present purposes, the term registration refers to the registration of individual objects (e.g., status object, IMO, RMO, specialized object) within a load balancing framework, including their creation (e.g., instantiation) from the object structures (e.g., classes) produced in the generation stage depicted in FIG. 4. In the monitoring stage, information is collected for the purpose of identifying a preferred server in accordance with a selected load balancing policy. In FIG. 5, state 500 is a start state.

Detailed Description Text (61):

As described above, the information to be collected may be determined by the selected load balancing policy, and will be used to identify a preferred server. In a present embodiment of the invention, the active policy for an application may be changed without disrupting the handling of client requests. Illustratively, this is done by temporarily pausing the operation of IMOs for the application, installing new status objects reflecting the new policy, then resuming the IMOs. Advantageously, the IMOs need not be altered or replaced.

Detailed Description Text (65):
In state 514, a specialized object is registered with the load-balancing framework (e.g., created from its object class). In state 516, parameters concerning the operation of the specialized object are set. Illustrative parameters include an identity of the RMO, the frequency of information retrieval from the RMO, an identity of the lookup table, method of interfacing with the RMO and/or lookup table, etc. In one embodiment of the invention, the specialized object analyzes the information collected from the servers hosting the application instances, identifies a preferred server in accordance with the load-balancing framework and updates the lookup table.

Detailed Description Text (68):
In state 520, a status object begins collecting or gathering information from its application instance. For example, where the selected policy favors the least-loaded application instance, a status object retrieves data concerning an instance's load (e.g., number of client requests or connected clients).

Detailed Description Text (72):
In one alternative embodiment of the invention, for example, clients access an instance of the application program directly (i.e., rather than connecting through a central server). In this alternative embodiment, the program instances exchange information (e.g., via status objects and/or other elements of a load-balancing framework) and redirect client requests as necessary to balance the requests in accordance with the selected policy.

CLAIMS:

38. A computer readable storage medium storing instructions that, when executed by a computer, cause the computer to perform a method for balancing requests for an application among a plurality of servers, wherein the requests are received at a central server, the method comprising:

selecting a policy for directing a request for the application to a preferred server, wherein said policy reflects a server factor for selecting said preferred server from the set of servers;

configuring a first status object to determine a first status of said server factor for a first instance of the application;

configuring a first server monitor object to receive said first status;

configuring a central monitor object to receive multiple statuses of said server factor for multiple instances of the application, including said first status;

examining said multiple statuses to select a preferred server; and

updating the central server to identify said preferred server;

wherein said first server-selection factor comprises an application-specific detail.

39. A method of load-balancing multiple requests for an application, wherein